C1. Introduction to Machine Learning

Advanced Machine Learning (MLA)

M2 Engineering of Intelligent Systems

& Advanced Systems and Robotics

2020-2021

Plan

1. A short survey of AI
   - AI: definition and history
   - Limitations and concerns
2. Basics of ML
   - Preliminary examples
   - ML tasks and types
3. The ML problems
   - Maths for numerical optimization
   - Inferring from a model
   - Learning a model
4. Why is it so hard to learn?
   - Memorisation vs. Generalization
   - Boundary decision
   - The curse of dimensionality
5. Model validation and good practices
   - Model assessment: estimation the generalization error
   - Hold-out, cross-validation, regularization
   - Common metrics and evaluation strategies

Pre-requisites

Mathematics

- Calculus, Linear Algebra

Numerical optimization

- Convex optimization, loss function, numerical gradient & Jacobian, gradient descent

Probabilities & statistics

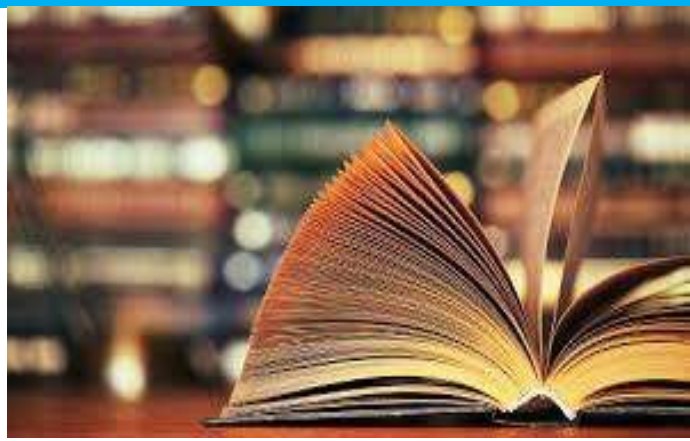
- Probabilities, random variables, statistical independence, conditional & joint probabilities, Bayes' theorem
- Mathematical expectation, statistical estimation, bias/variance

And more

- Entropy, mutual information, ...

Holy books

Books & papers



Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning - Data Mining, Inference, and Prediction (Springer, 2009)

Bishop C., *Pattern Recognition and machine learning* (Springer, 2006)

Goodfellow, I., Bengio, Y. & Courville, A., *Deep Learning. (MIT Press, 2016)*

More readings on the web

Some blogs on ML

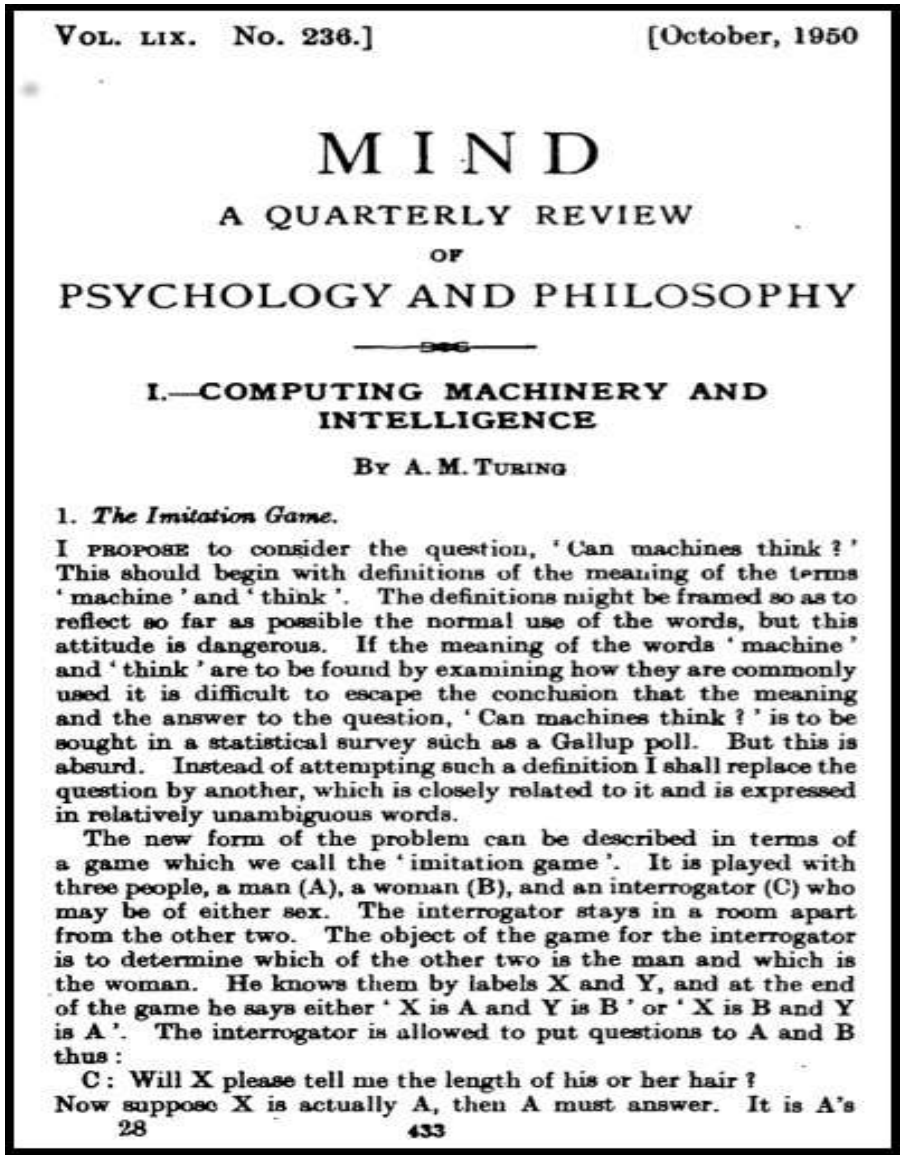
Computer vision for dummies for the illustration of the « curse of dimensionality »

SCIENCES SORBONNE UNIVERSITÉ

1.
A short survey of AI

What is AI?

- Artificial intelligence
  - Machines capable of **simulating human intelligence**
  - **1950** : Turing test (or imitation game)
  - **1956** : Dartmouth Summer Research Project on Artificial Intelligence with J. McCarthy, C. Shannon, N. Wiener, Marvin Minsky, etc...
  - Mainly **symbolic AI** : symbolic data, and manually defined rules



VOL. LIX. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING

1. *The Imitation Game.*

I PROPOSE to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the 'imitation game'. It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A'. The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair?

Now suppose X is actually A, then A must answer. It is A's

28 433

What is AI?

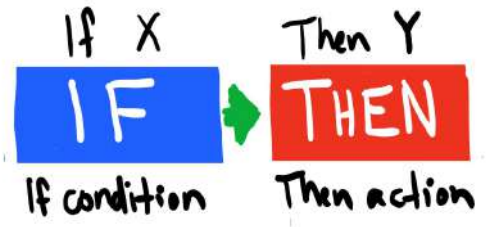
- Artificial intelligence

"After WWII, a number of people independently started to work on intelligent machines. The English mathematician Alan Turing may have been the first. He gave a lecture on it in 1947. He also may have been the first to decide that AI was best researched by programming computers rather than by building machines. By the late 1950s, there were many researchers on AI, and most of them were basing their work on programming computers."

(John McCarthy)

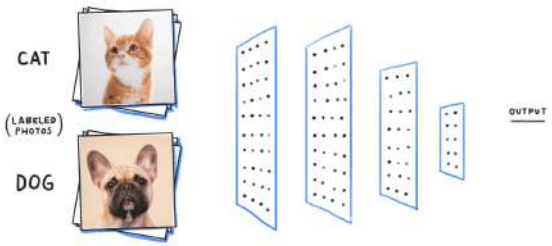
What is AI?

Rules



Data

Decision



Symbolic AI
(deductive)

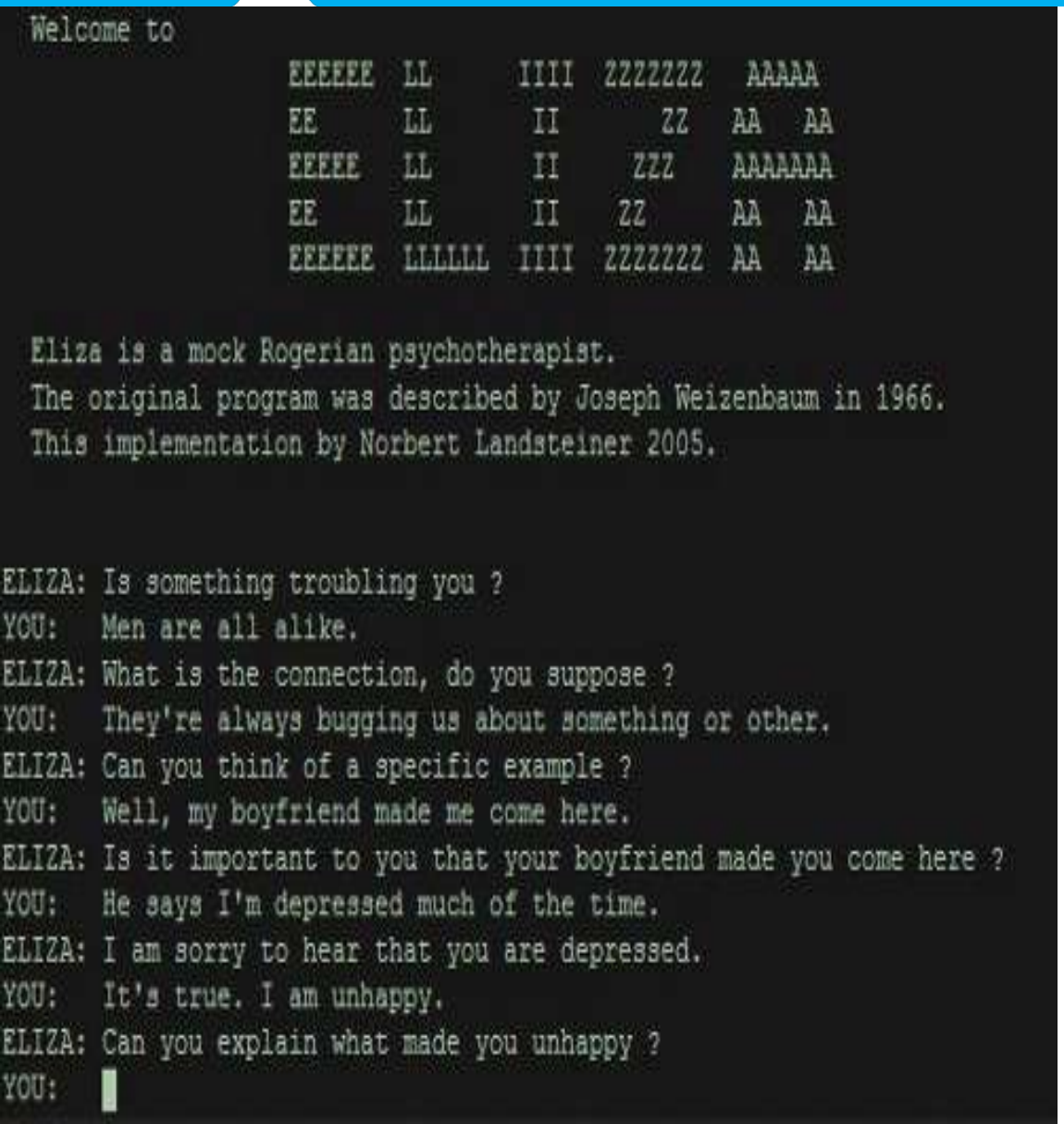
Human knowledge, abstraction & reasoning

« Embodied » AI
(cybernetics, robotics)

Connectionist AI
(inductive)

ML, perception

What is AI?



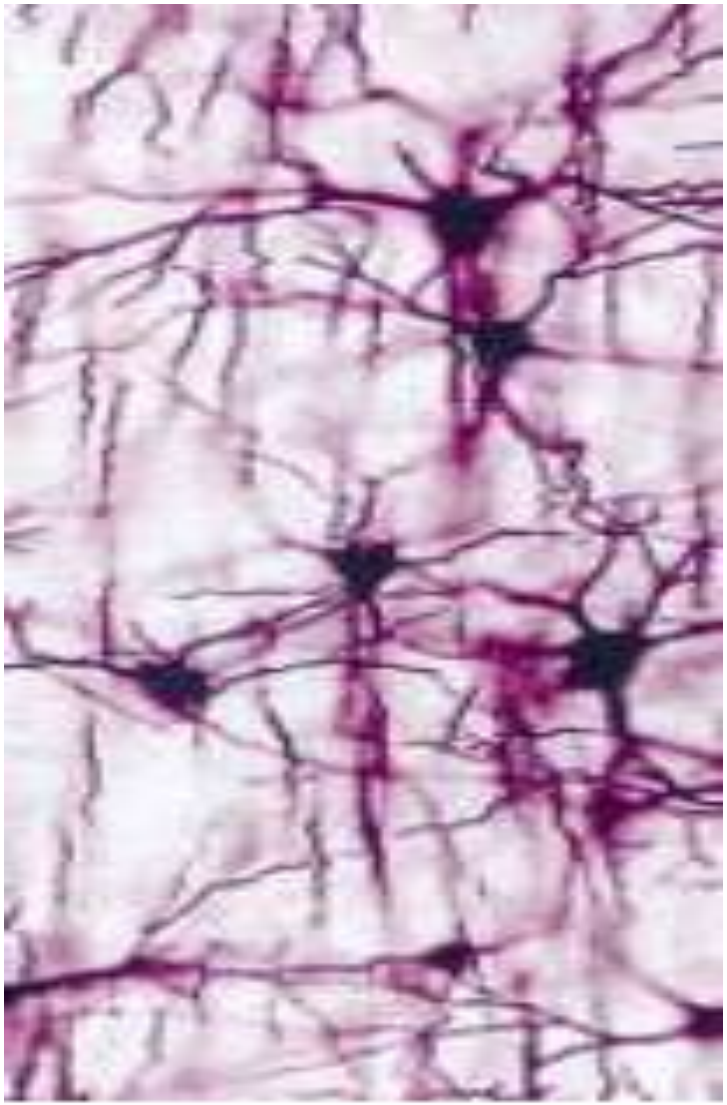
Symbolic AI

- Explicitly represents **human knowledge and reasoning** in a symbolic & declarative form

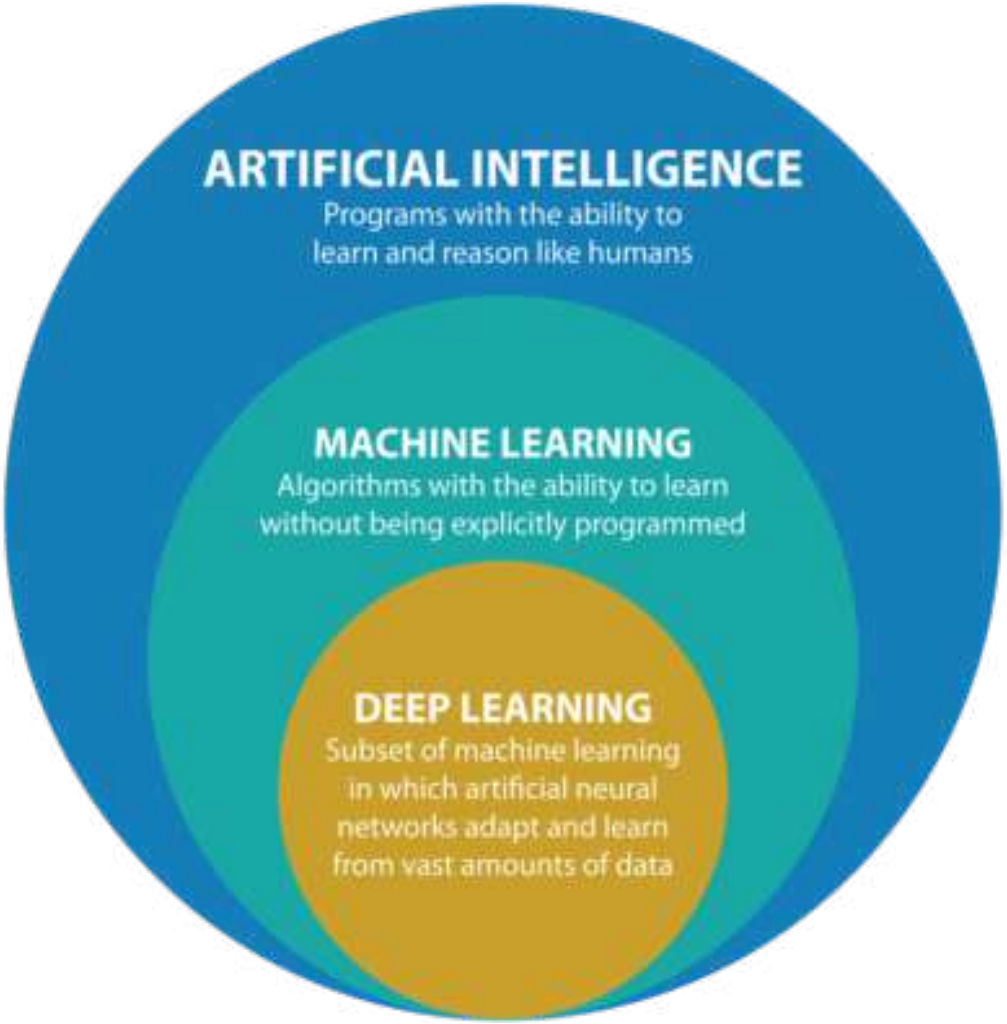
- Deductive reasoning, logical inference, search algorithms
- Natural language processing and checkers (50's), expert systems (1980's)

What is AI?

- Connectionist AI
  - Simulating the functioning of **human brain** and **cognition**
  - Processing information from perception (sensors) to cognition (representation and decision)
  - **1954**: first **artificial neural network** (B. Farley & W. Clark , MIT)
  - Can be **trained** from data
  - Computer vision, cybernetics



ML as a branch of AI



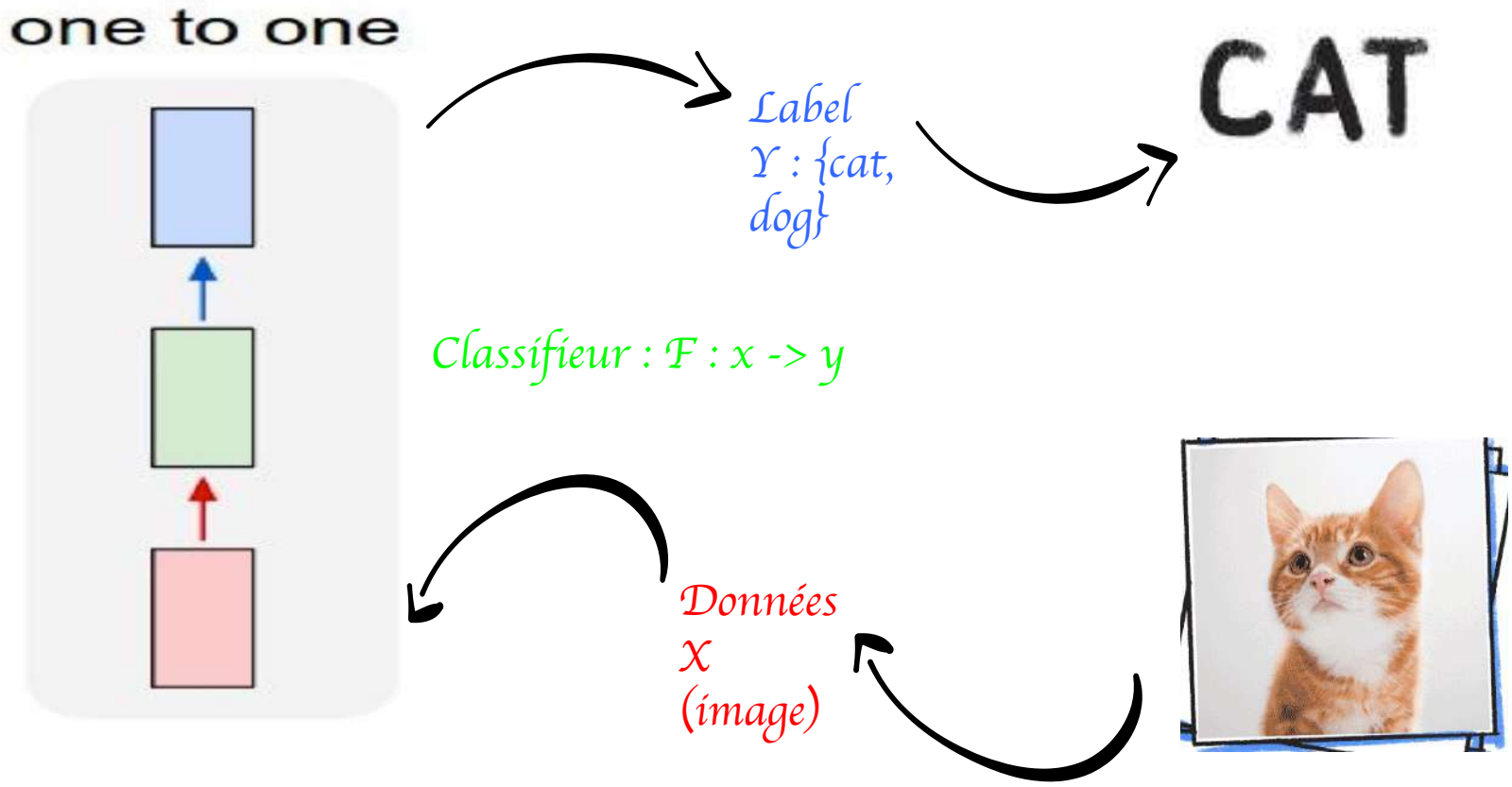
Le ML est la branche de l'IA qui s'occupe de :

- résoudre des tâches définies explicitement
- par apprentissage automatique de règles
- à partir de données

Ces 3 parties impliquent l'intervention d'un humain

Definition

The « black box » definition



Limitations of AI

Reliability and trustability: chiwawa or muffin?

Limitations of AI

Reliability and trustability: panda+noise =gibbon?

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

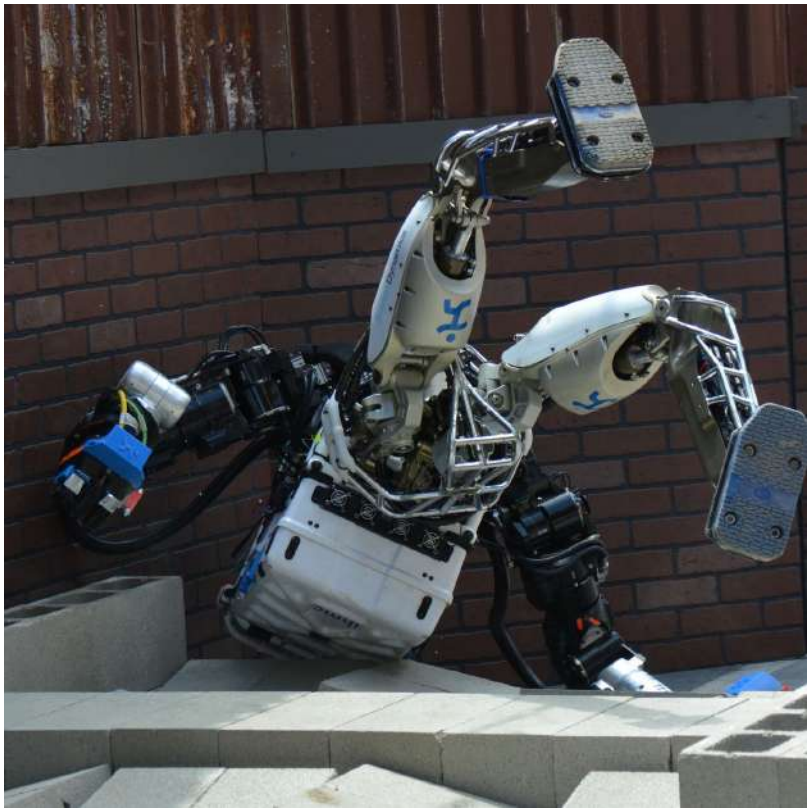
The Moravec paradox

AI can solve more easily complex tasks than simple tasks (from human point of view)

- Complex: reasoning, etc...
- Simple: sensori-motor tasks (walking, etc...)

Tasks which appear to us elementary have been perfectioned during million years of human evolution

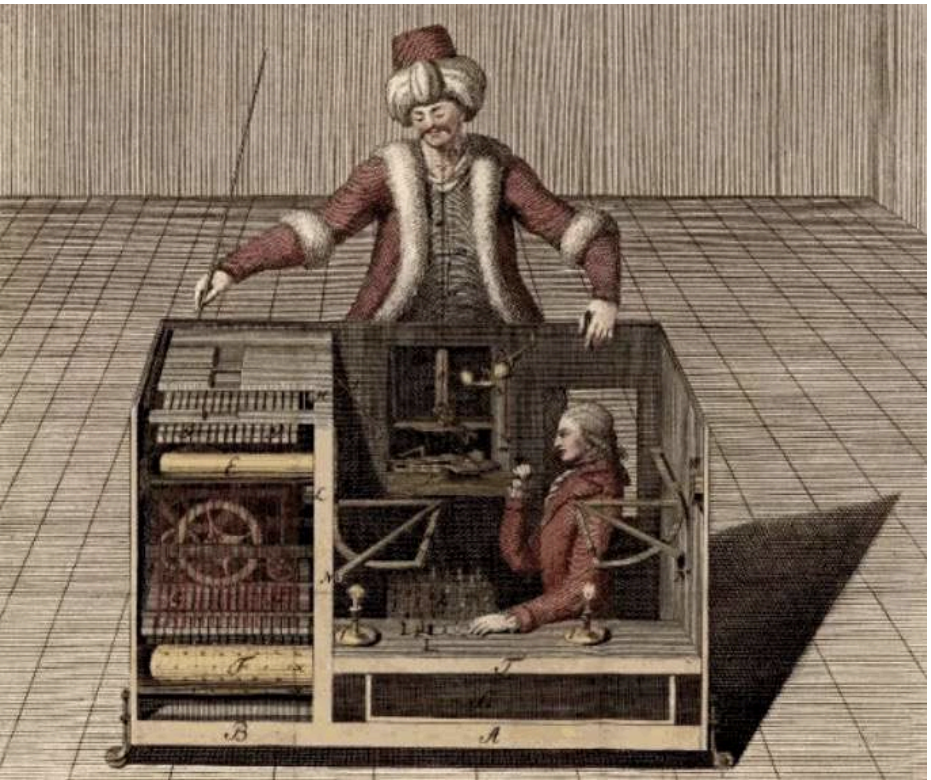
In fact: perception/action coordination is extremely complex!



No need for humans?

Behind AI: many humans!

Listen on France Culture : [Les "travailleurs du clic", ces humains cachés dans les machines]



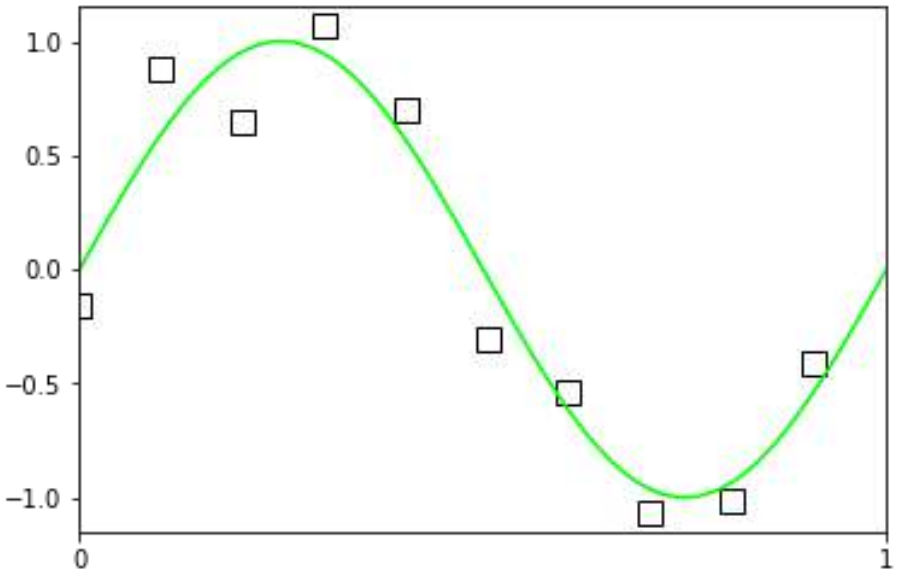
Singularity ?

We are still far away from the « strong » AI (cognitive, social, etc. functions)

SCIENCES SORBONNE UNIVERSITÉ

2.
Basics of ML

Simple example (1)

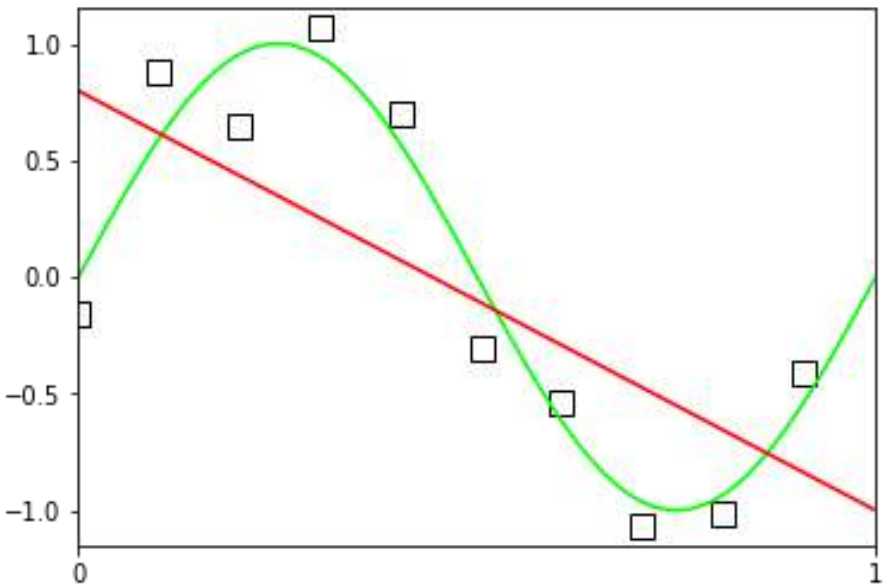


- Curve fitting
- How can we optimally determine the unknown function from a limited number of observations?
- The hidden function is a sinus function
- The observations (*N=10*) are limited and noisy samples (gaussian noise)
- This is a typical **regression problem**

Simple example (1)

- A simple solution is to assume the family of the function, e.g., **polynomial function**

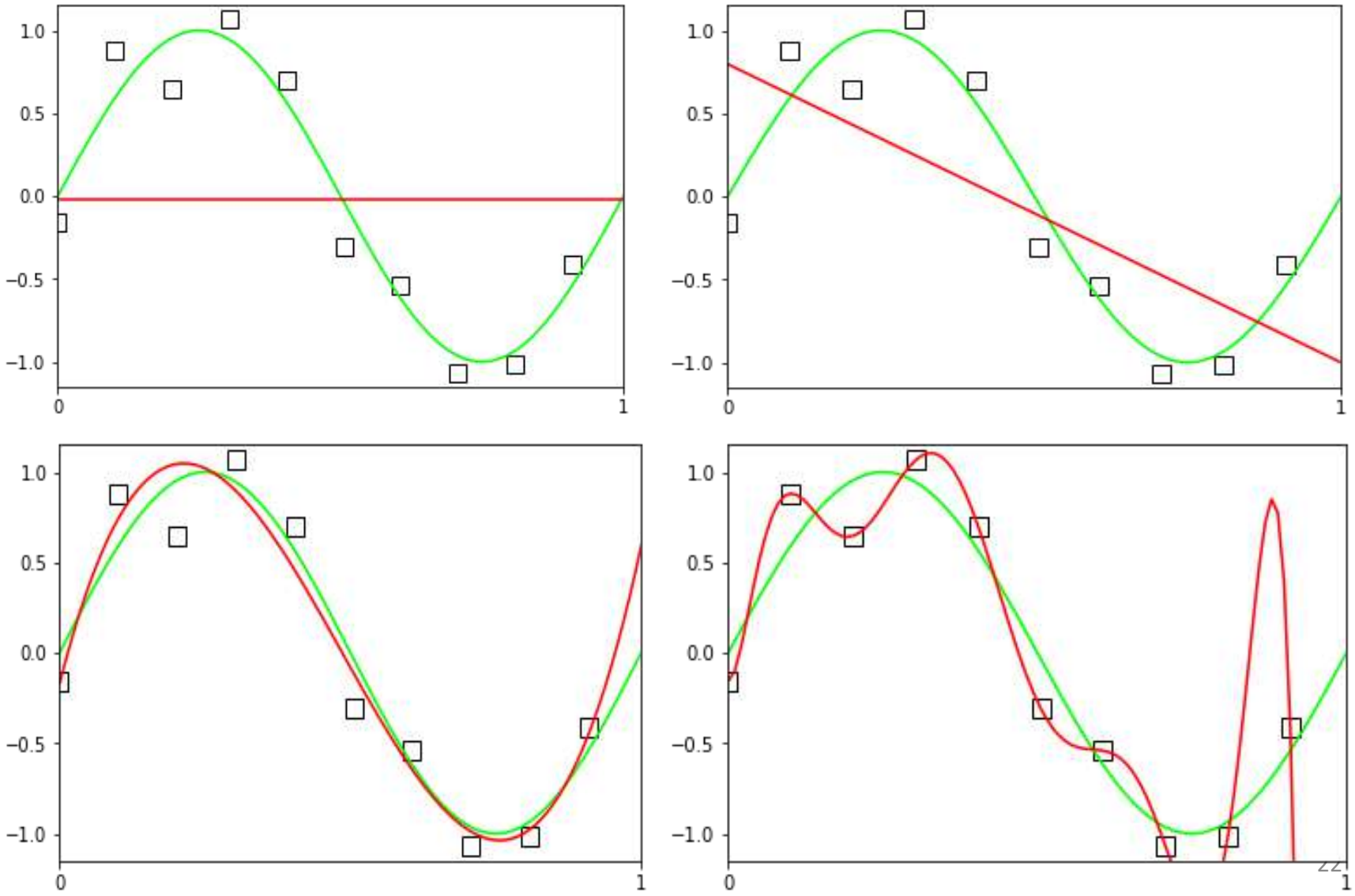
$$y = \sum_{i=0}^{N} w_i x^i$$



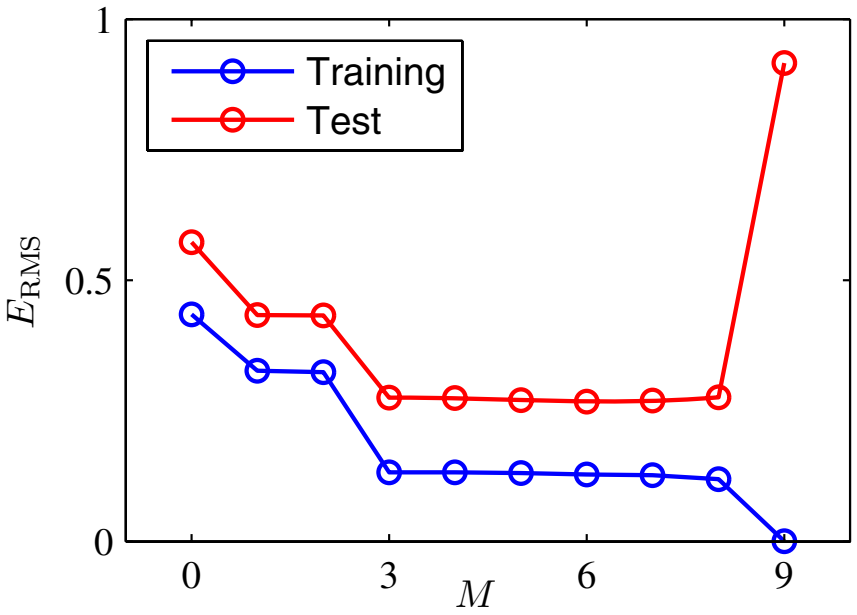
- Define a loss that describes the error measure between the true and the estimated function: e.g., **sum of squared errors**

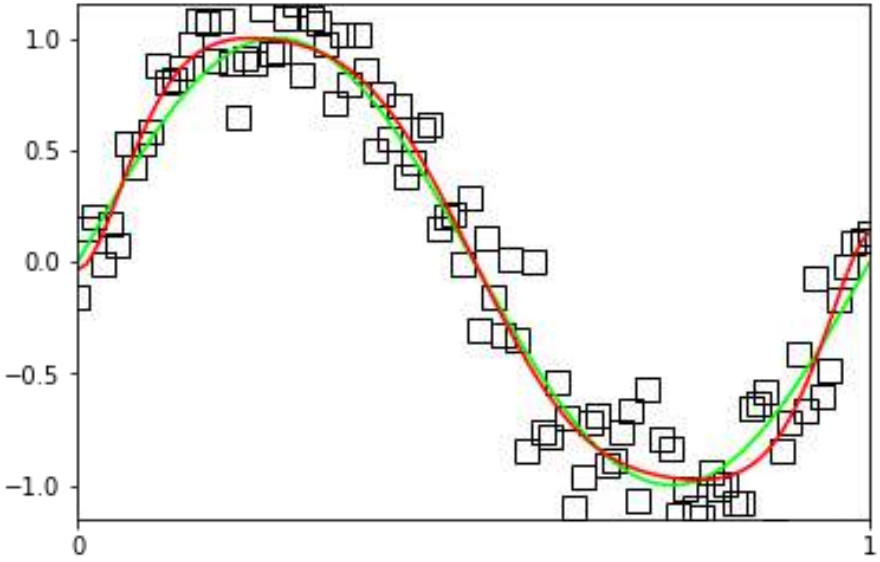
- Polynomial regression with SSE is a **linear problem** in the unknown parameters space $\{w_1, \ldots, w_N\}$
- Convex optimization with a **unique solution**

Simple example (1)



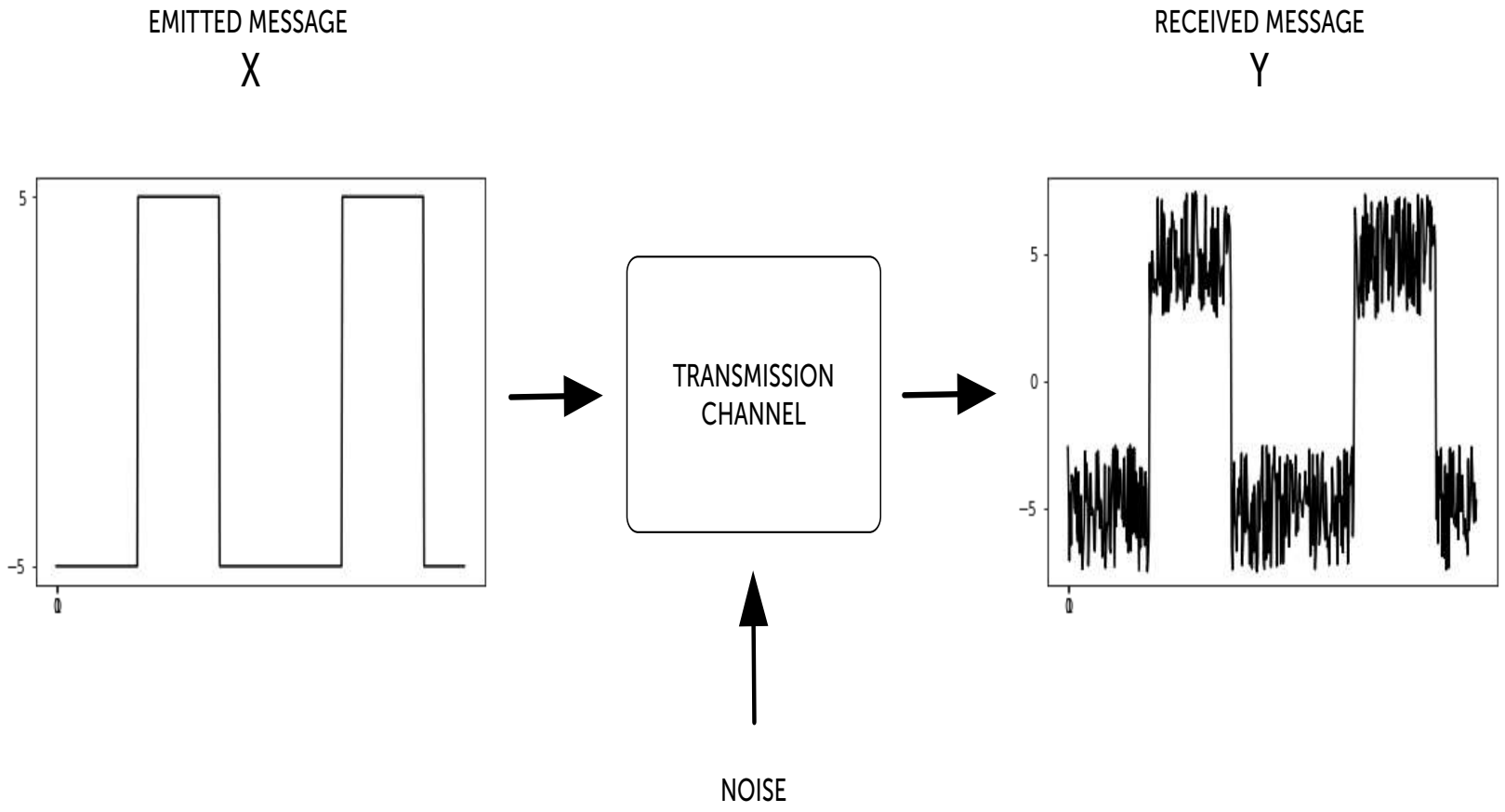
Simple example (1)





Simple example (2)

A classic problem from communication theory



Simple example (2)

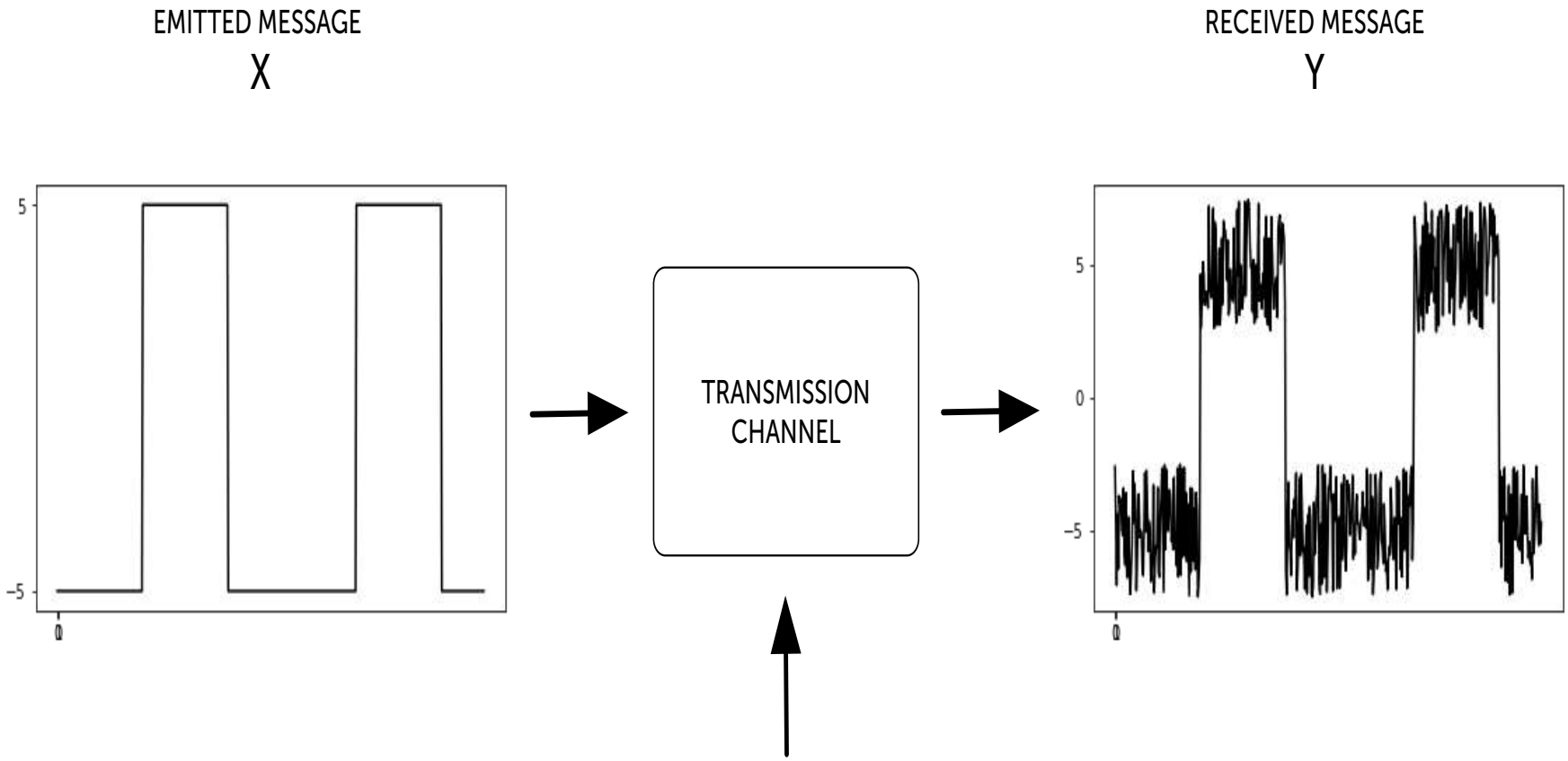
- Suppose that we have a **random source X** emitting a message **x** and that the destination receives the message **y**, as a noisy version of x
- How can we optimally retrieve the emitted message x from y ?

- Y is the **observed variable**, with continuous values
- X is the **hidden variable**, with states in {0, 1}

- X is generally not observable, so we try to **infer** it from the observation Y (inverse problem)

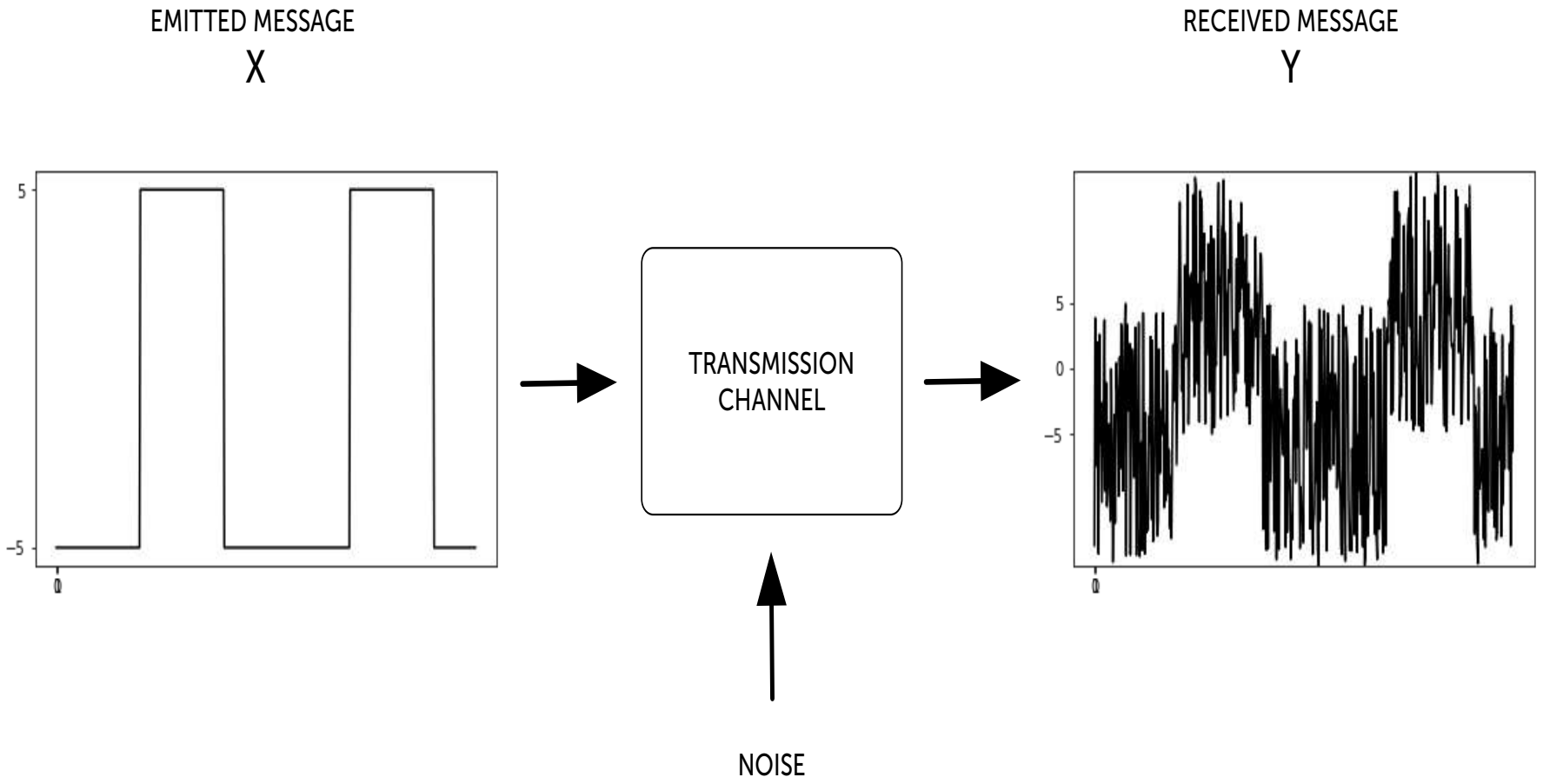
- This is a typical **classification problem**

Simple example (2)



– Intuitive solution : threshold Y on 0

Simple example (2)



– Not always that simple!

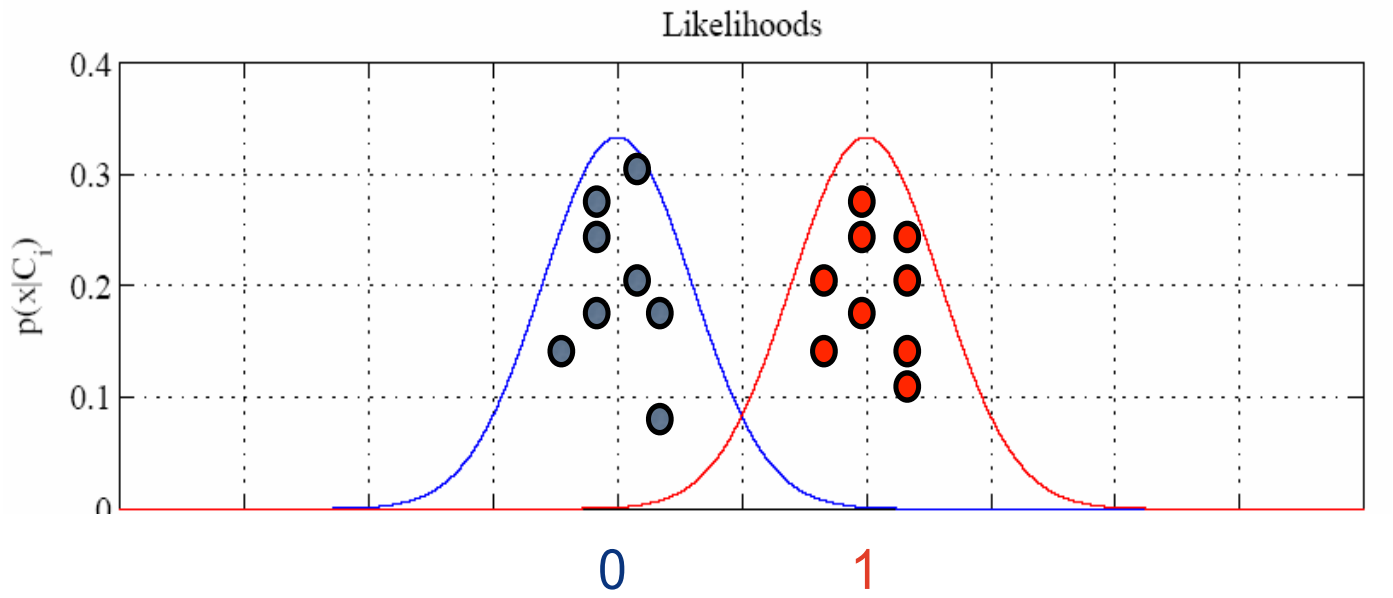
Simple example (2)

Solution 1

– Learn the distribution that **generate** Y if I know X

$$P(Y \mid X = x)$$

- Define a decision function that predicts the state of X from the observation y

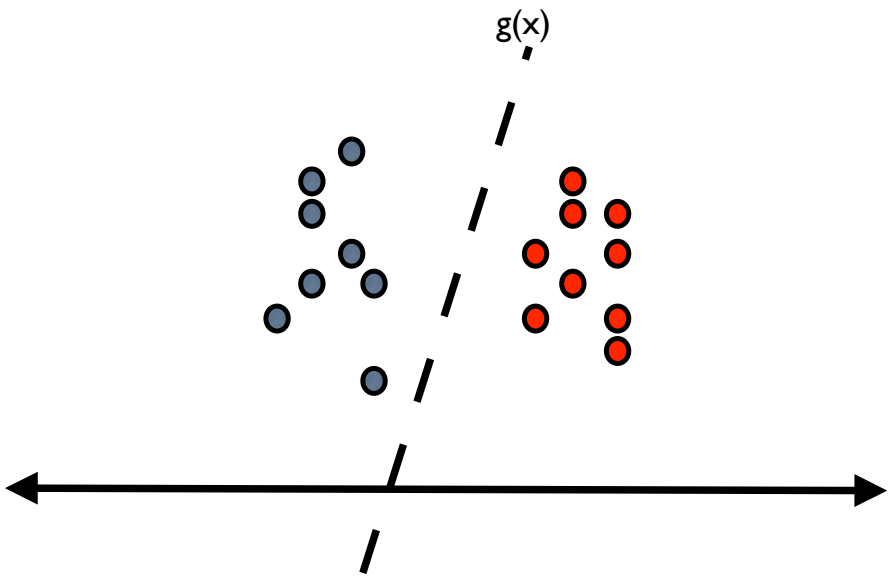


Simple example (2)

Solution 2

– Determine the optimal boundary that can best **discriminate** the states of X from the observed data Y

– Define a decision function as a +/- distance from this boundary



Basics concepts

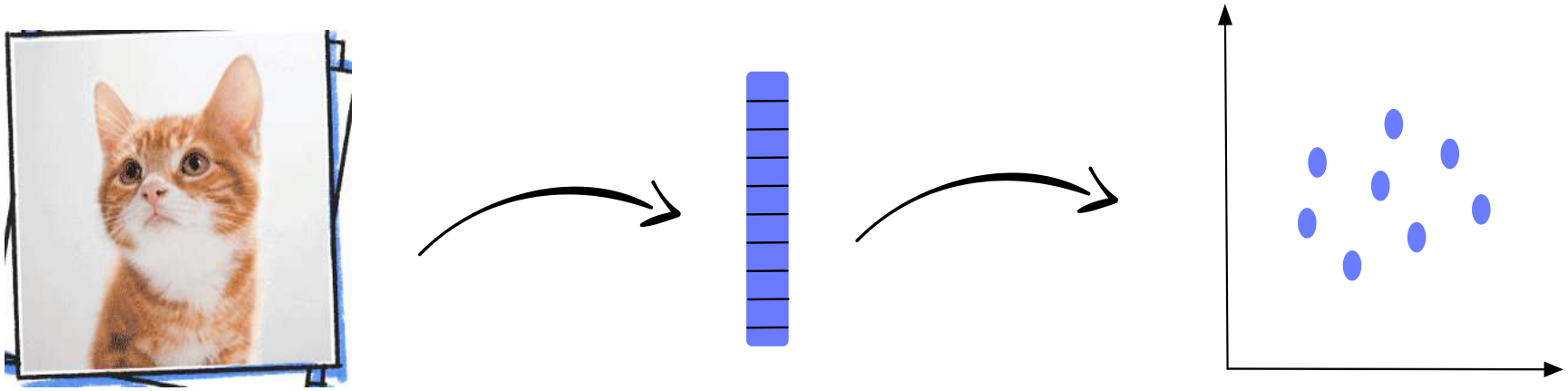
1. Data

One **instance** which provides a **measurable information** about the state of an object, a system, or an individual

Examples: physical quantities, image, text, user's preferences

2. Feature

A **representation** of all or part of the information contained in the raw data

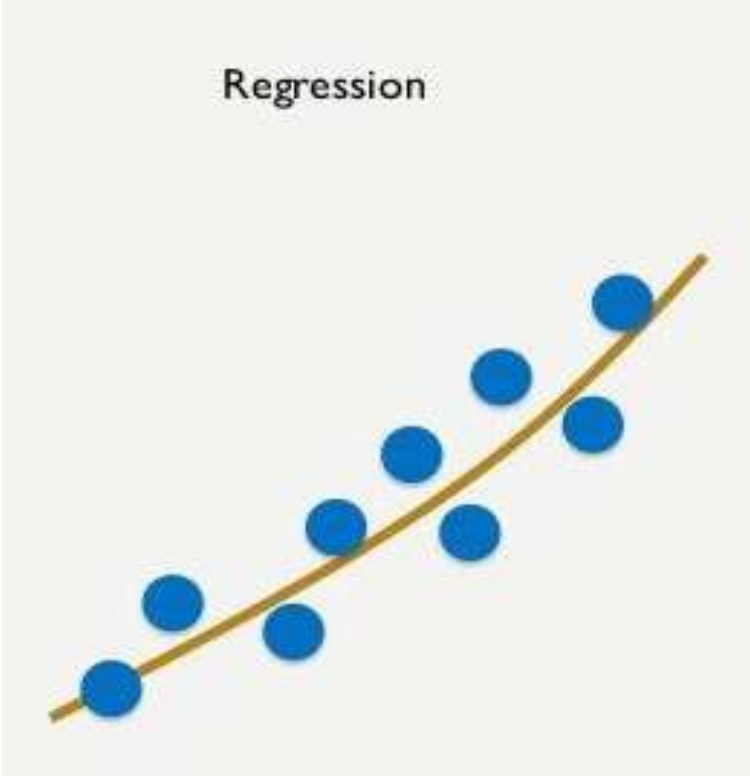


donnée brute vecteur de *feature* espace de données

ML tasks

1. Regression

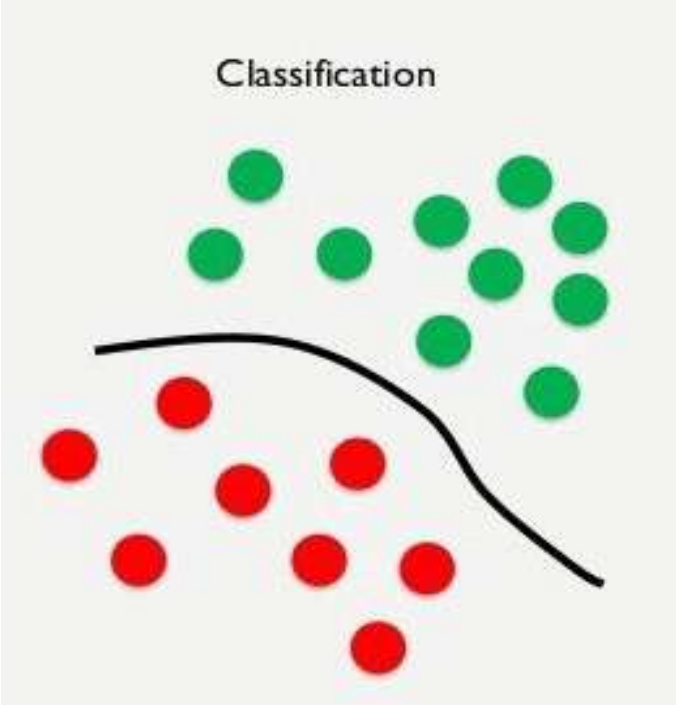
Predict a numerical value from a set of continuous/symbolic data
Example: curve fitting, trends forecasting, fault prediction, etc…



ML tasks

2. Classification

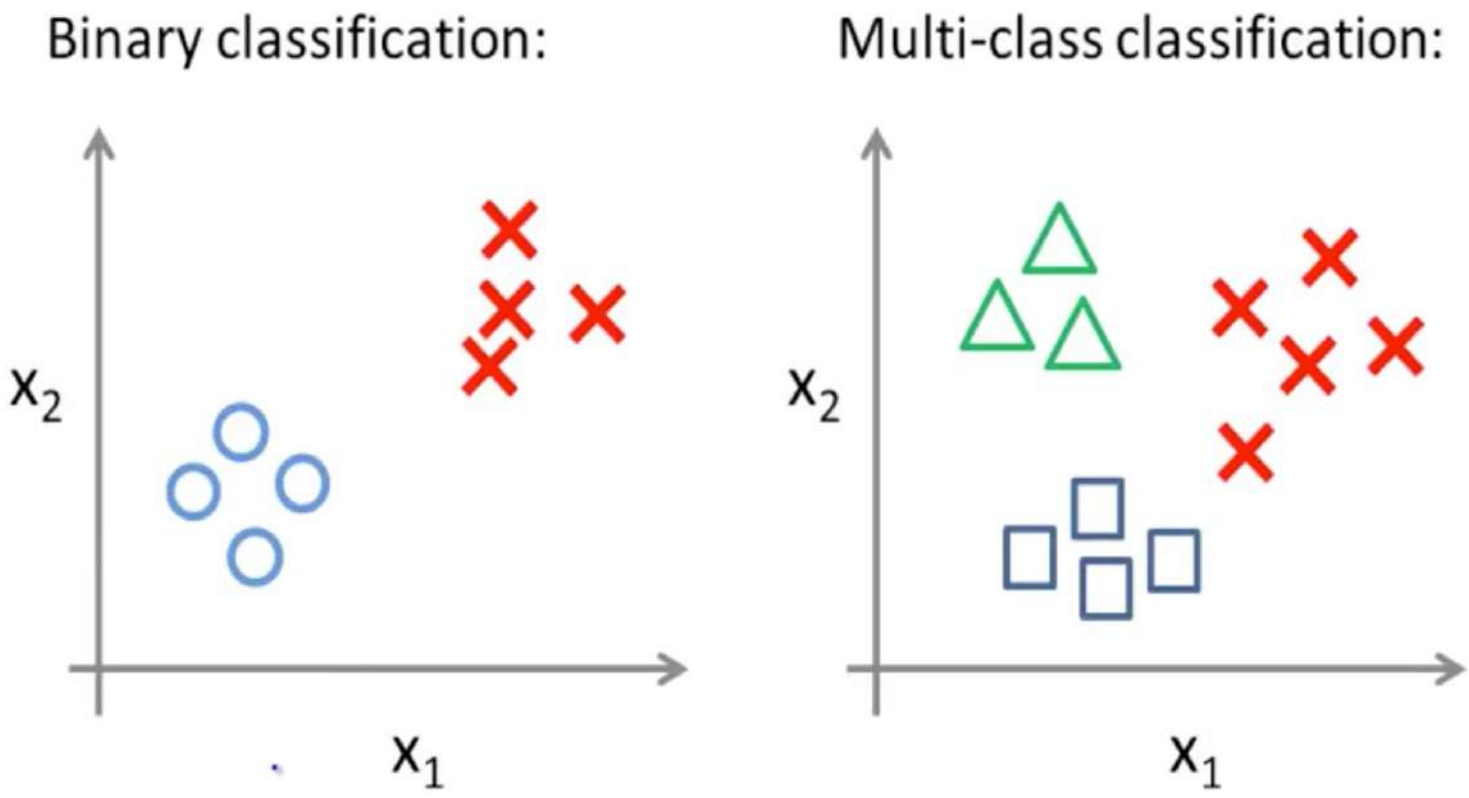
Predict the belonging of a data to a given class
Exemples : object recognition, sentiment analysis, etc...



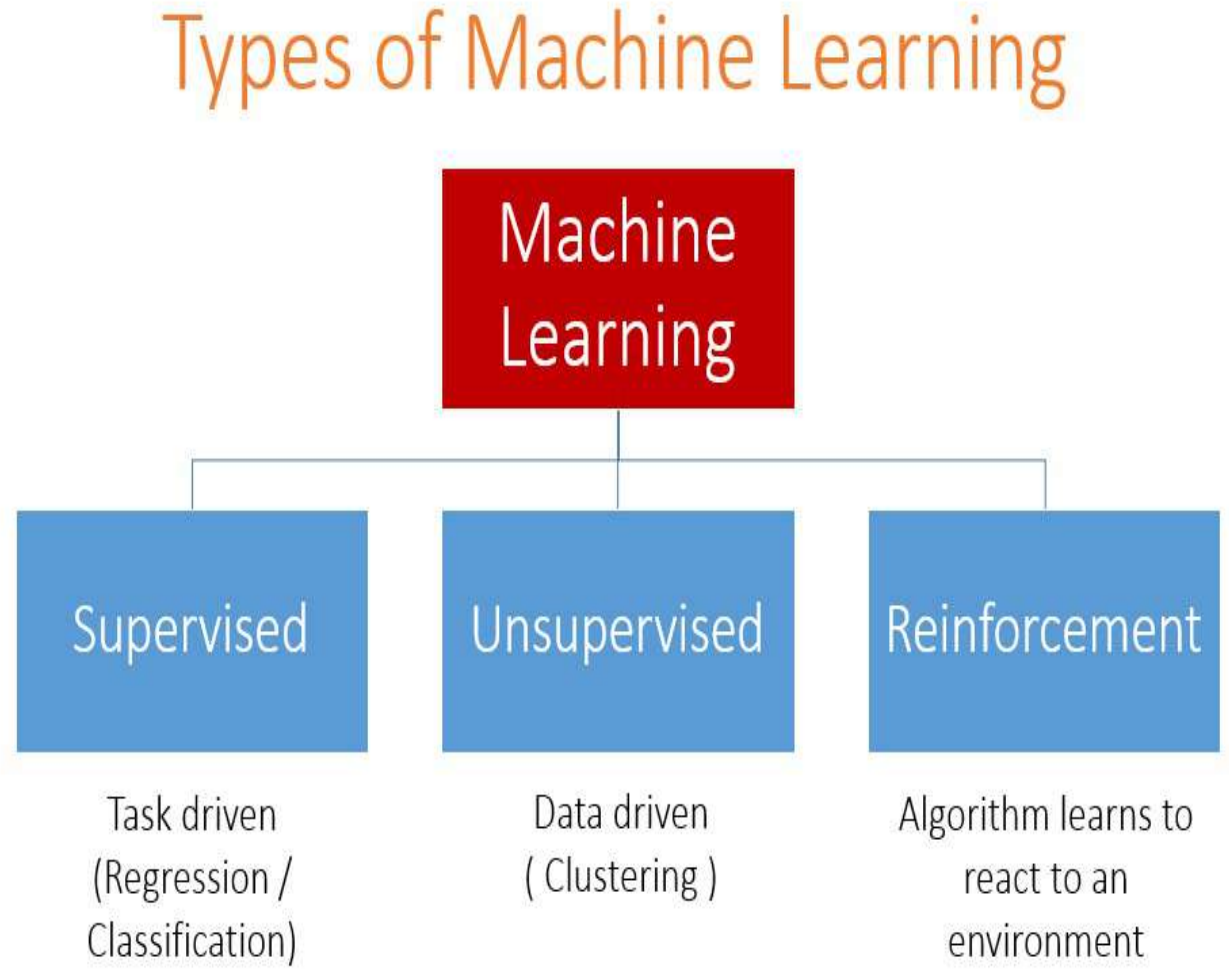
ML tasks

2. Classification

Binary (one-vs-one, one-vs-all), muti-class, multi-label



Types of ML



1. Supervised learning
Learning guided by the **task**

2. Unsupervised learning
Learning guided by the **data**

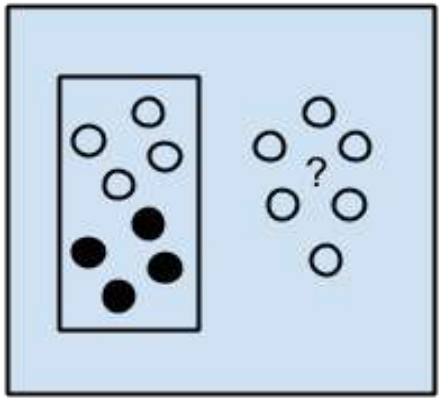
3. Reinforcement learning
Learning by **trial & error**

Types of ML

1. Supervised learning

During learning, one can see the data and the labels

- Pros: learn explicitly the task to solve
- Cons: requires annotated data (by humans), time-consuming…



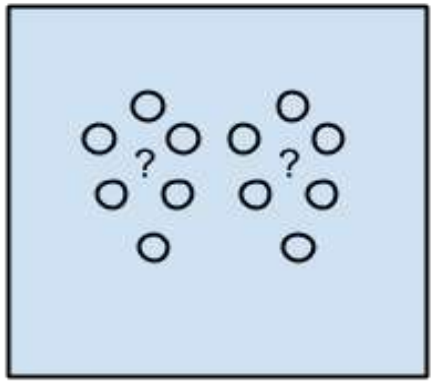
Supervised Learning Algorithms

Types of ML

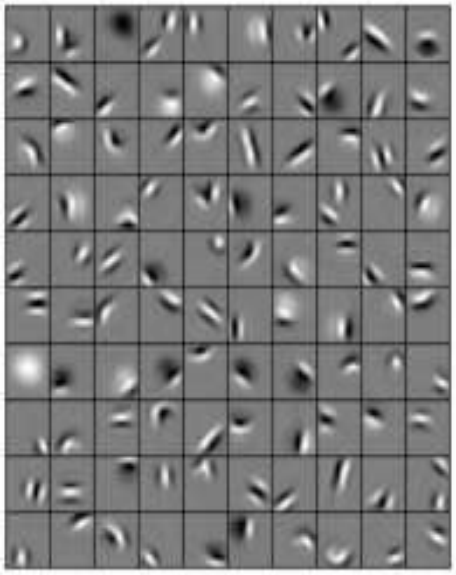
2. Unsupervised learning

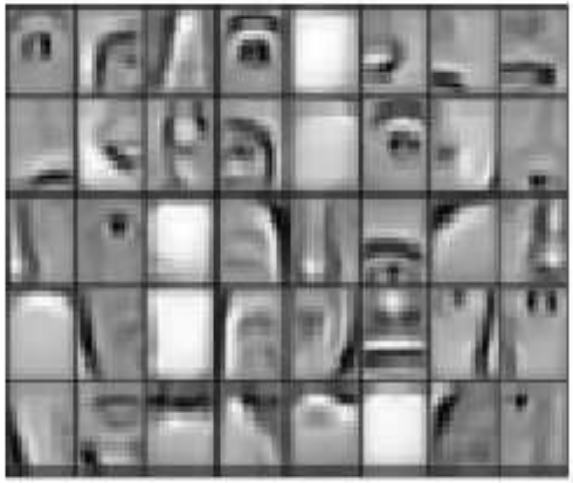
Can only see the raw data

- Pros: possible to learn from large amount of data
- Cons: interpretation



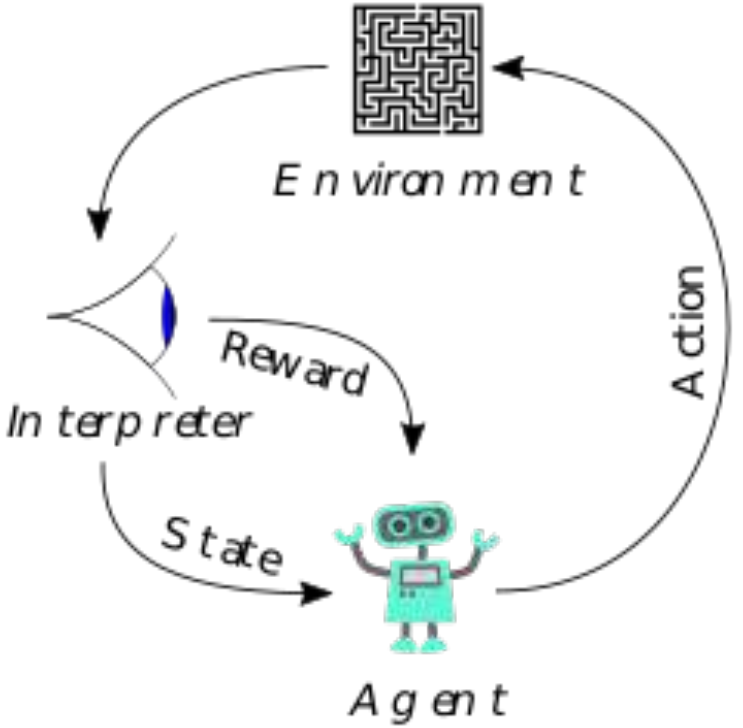
Unsupervised Learning Algorithms





Types of ML

3. Reinforcement learning (« trial & error »)
- Pros: on-line/interactive learning, numerical simulation, adaptation to the environment
- Cons: rather for exploration/adaptation tasks, definition of the reward



SCIENCES SORBONNE UNIVERSITÉ

3.
The ML problems: Learning & inferring

Learning as a numerical optimization problem

One can formulate a ML problem as a numerical optimization problem whose objective is to minimize (conversely maximize) a loss function L defined w.r.t. a given task.

One want to estimate a function that maps the data x to the decision y,

$$\mathcal{F} : x \rightarrow y$$

One define a loss function between the actual output and the predicted one:

$$\mathcal{L}_{\hat{\mathcal{F}}}(y, \hat{\mathcal{F}}(x))$$

We want to estimate optimally the decision function from the available data,

$$\hat{\mathcal{F}} = \arg\min_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$$

Usual loss functions

There exists a large variety of loss functions, depending on the nature of the task and the statistical/optimization framework

Regression
- Sum of squared errors,
- MSE : mean squared error, MAE : mean absolute error
- Log-likelihood (Bayesian optimization)

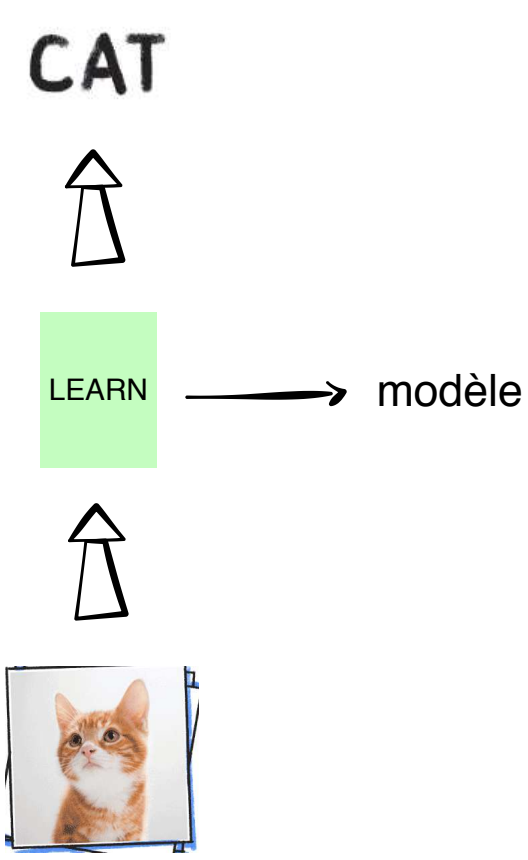
Classification
- Logistic loss
- Hinge loss (SVM)
- Cross-entropy loss
- etc...

In ML, this loss function are generally (but not necessarily) continuous and differentiable

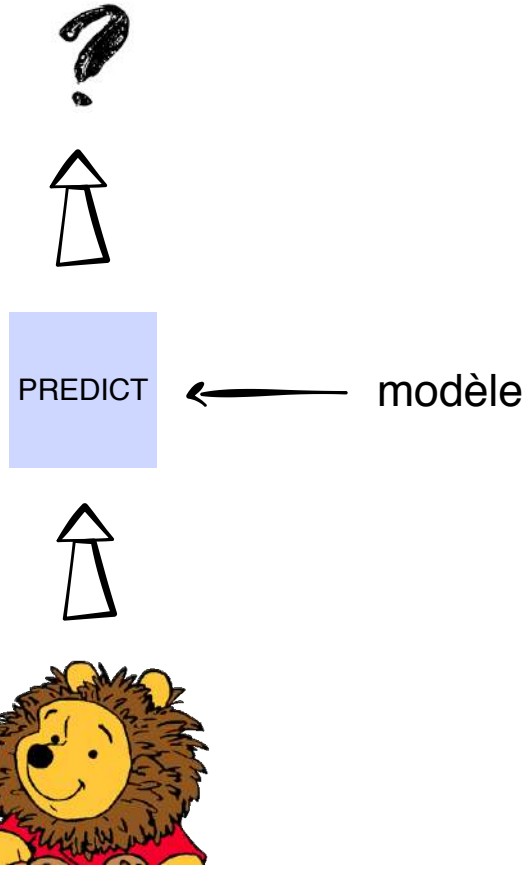
Learning and inferring

The two problems of ML

1. **Learning** a model given the data,



2. **Inferring** from a model

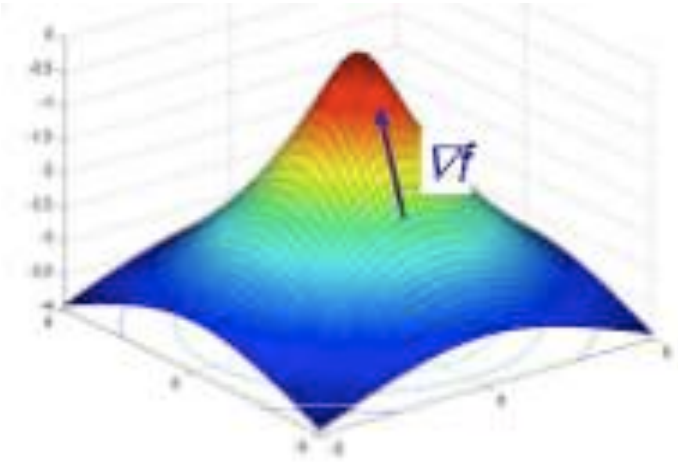


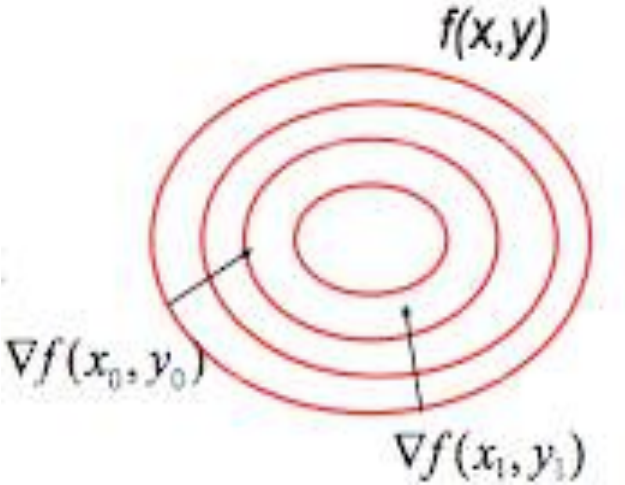
Maths for numerical optimization

- Gradient of a function

Definition 1 *If f is a function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^1$, then the* **gradient** *of f is defined as:*

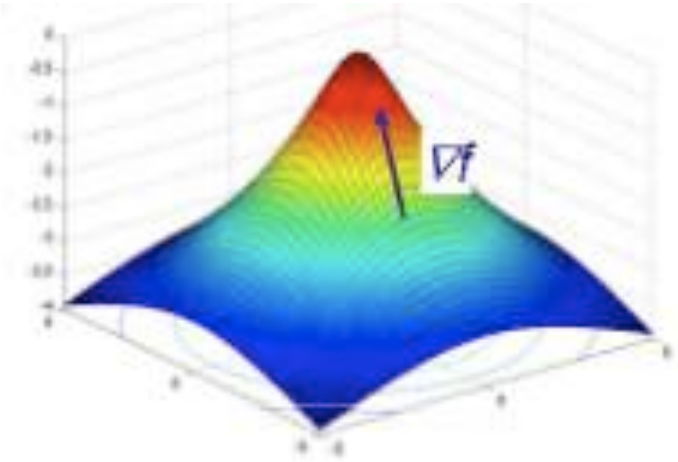
$$\nabla_x f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^\top$$

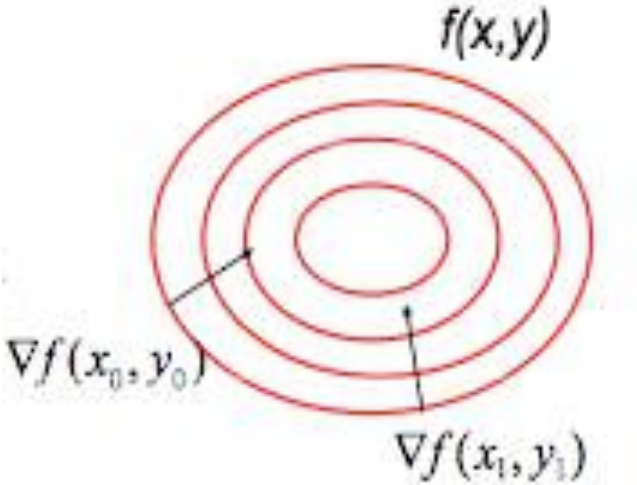




Maths for numerical optimization

- Gradient of a function
  - Points the direction of **maximum increase** of the function
  - Perpendicular to the contour where the function is constant





Maths for numerical optimization

- Critical point
  - If the gradient is null at a point, then:
    - No direction of **increase** or decrease
    - **Local min** or **max**, or **saddle** point
  - If at a local min/max/saddle point, then:
    - Gradient is null

Definition 2 $x_0 \in \Omega \subset \mathbb{R}^n$ *is a* **critical point** *of a function* f *if and only if,*

$$\nabla_x f(x_0) = 0$$

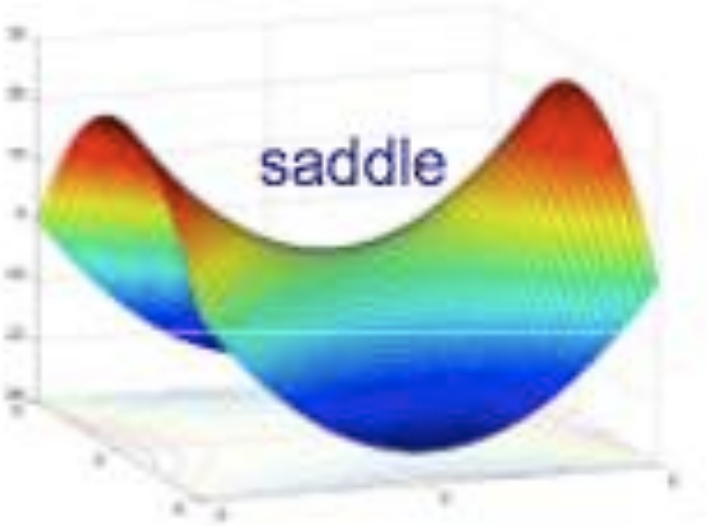
Maths for numerical optimization

- Hessian of a function

Definition 3 *If f is a function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^2$, then the* **Hessian** *of f is a symmetric matrix defined as:*

$$H_{i,j} = \frac{\partial^2 f}{\partial x_i \, \partial x_j}$$

  - Second order partial derivatives of a function
  - Indicates the curvature of the function at a point

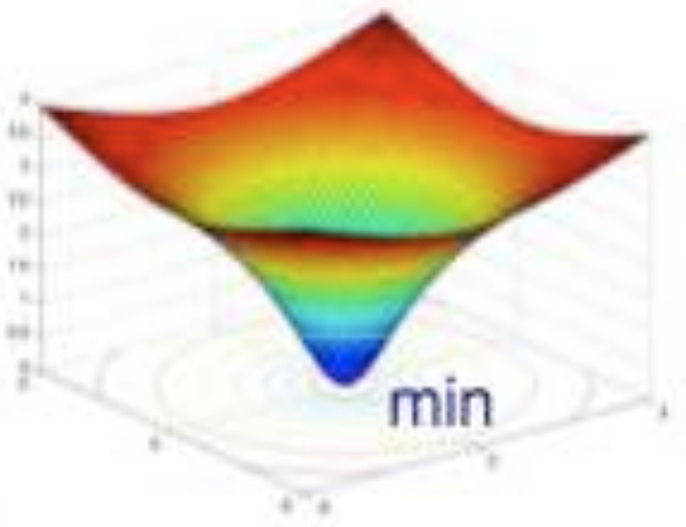


Maths for numerical optimization

- Convexity and global minimum of a function

Proposition 1 *A function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^2$ if* **convex** *if and only if the Hessian of f is positive definite (i.e., the eigen-values of the Hessian matrix are positives)*

Proposition 2 *If $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is* **convex**, *then f admits at least one minimizer, and one and only one if f is* **strictly** *convex*

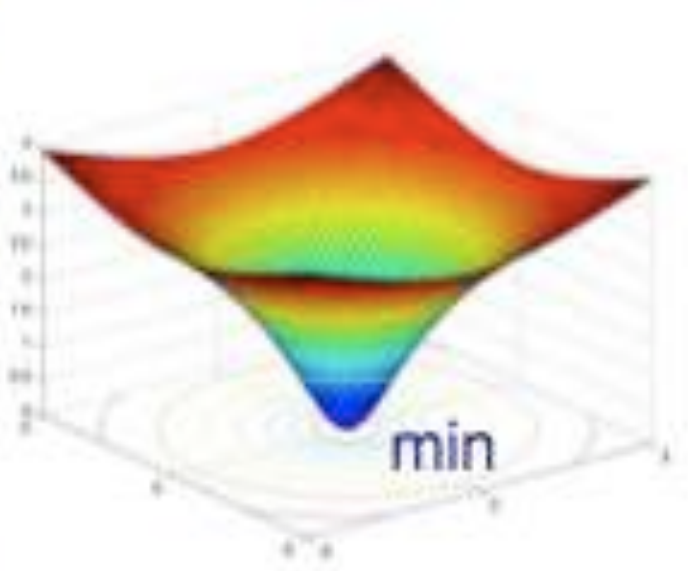


Maths for numerical optimization

- Optimization of a differentiable function

Definition 4 *The general problem for the optimization of differentiable functions is:*

$$\min_{x \in \Omega} f(x)$$



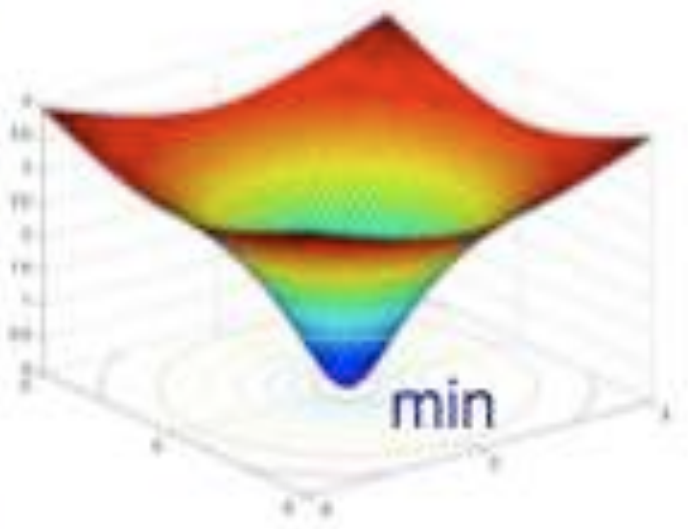
Maths for numerical optimization

- Optimization of a differentiable function

Proposition 3 *If f is convex, and if Ω is convex and closed, then f admits one and only one minimum.*

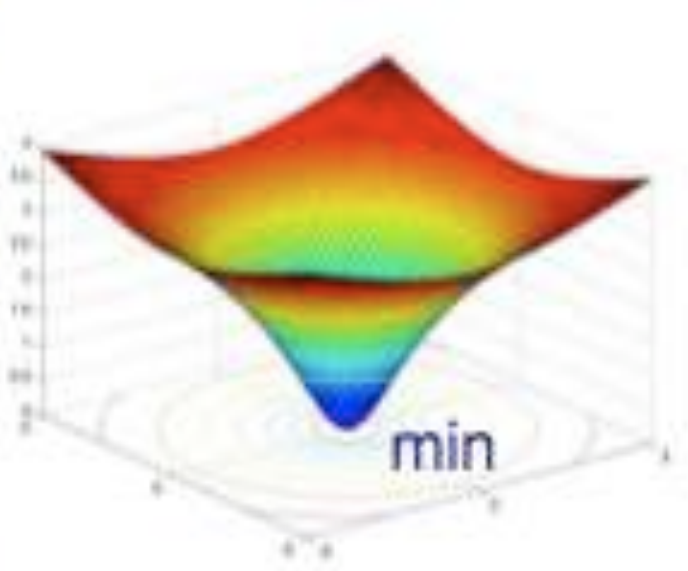
Proposition 4 *The critical points $x*$ of a differentiable and convex function f are global minimas of f*

$$\nabla_x f(x*) = 0$$



Maths for numerical optimization

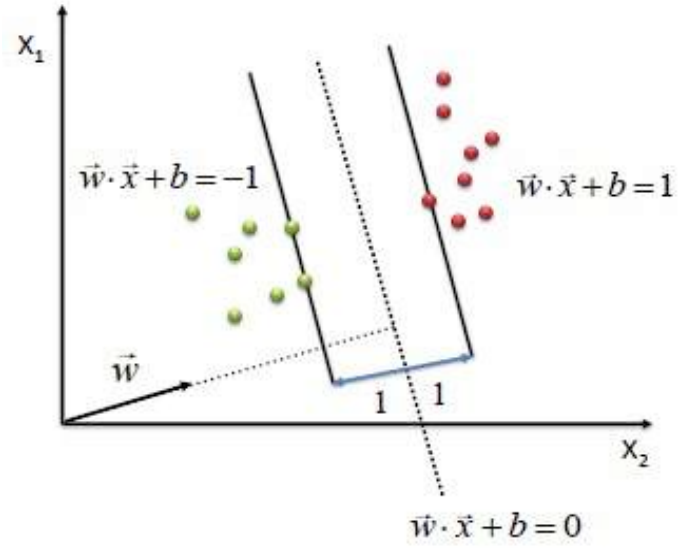
- Convex optimization
  - If *f* is a function from R to R: dichotomy, Newton, gradient descent
  - If *f* is a function from R^N to R^N: gradient descent



Maths for numerical optimization

- Beyond the simple case
  - Optimization **under constraints**: KKT conditions and Lagrange multipliers, primal/dual optimization
  - Optimization of **non-differentiable** functions

➢ SVM is an example of optimization under constraint



$$\max \frac{2}{\|w\|}$$

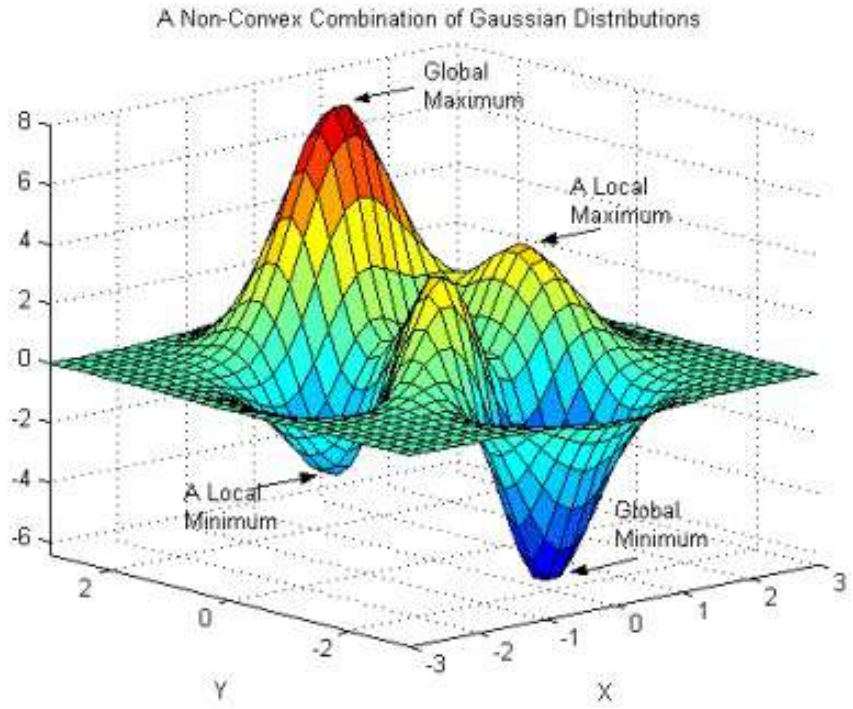
s.t.

$(w\cdot x+b)\geq 1, \forall x$ of class 1

$(w\cdot x+b)\leq -1, \forall x$ of class 2

Maths for numerical optimization

- Beyond the simple case
  - In real-world problems, functions are usually **non-convex**
  - Multiple local minima ☹



Inferring from models

- Bayesian inference

Definition 1: 1-D Gaussian distribution

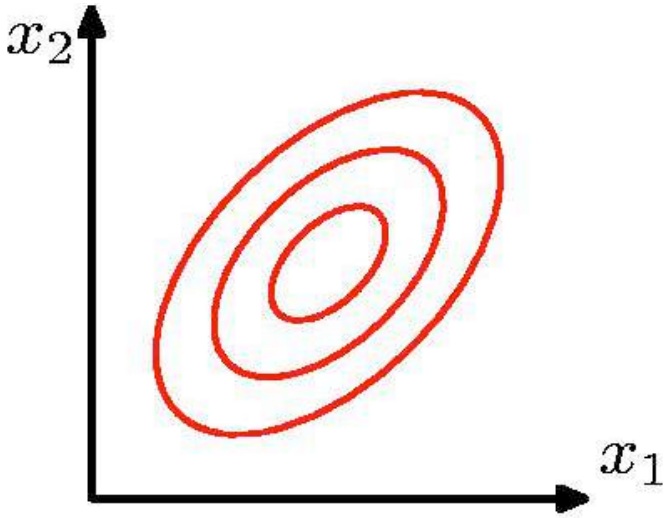
$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Property 1:

$$\mathcal{N}(x|\mu,\sigma) > 0$$

Property 2:

$$\int_{-\infty}^{+\infty} \mathcal{N}(x|\mu,\sigma) = 1$$



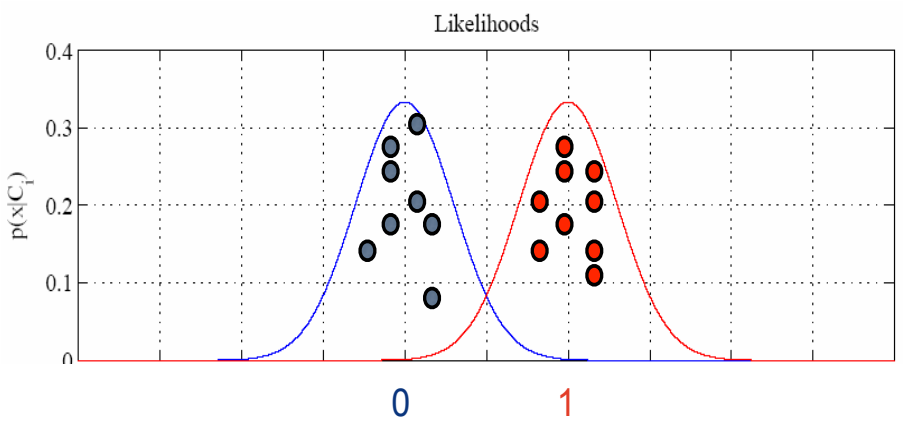
Definition 2: N-D Gaussian distribution

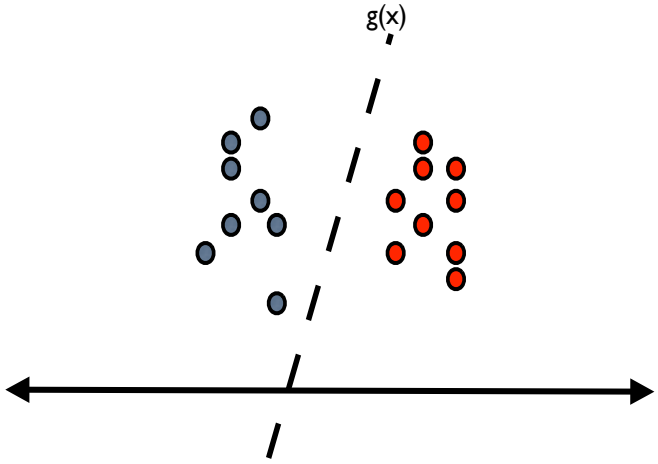
$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

Inferring from models

- The inference problem
  - Generative models
    - Learn: $p(s, \boldsymbol{x}) = p(\boldsymbol{x}|c)p(s)$
    - Infer: $p(s|\boldsymbol{x}) = \dfrac{p(\boldsymbol{x}|s)p(s)}{p(\boldsymbol{x})}$

    (Bayes rule)
  - Discriminative models
    - Directly learn: $p(s|\boldsymbol{x})$





Inferring from models

- Maximum a Posteriori (MAP) estimate
  - The optimal value is determined as the one maximizing the conditional probability,

$$s^* = \underset{s}{\operatorname{argmax}}\, p(s|\boldsymbol{x})$$

  - Then, we need to know,

$$p(s|\boldsymbol{x})$$

  - But we actually learned,

$$p(\boldsymbol{x}|s)$$

Inferring from models

- Maximum a Posteriori (MAP) estimate
  - Applying Bayes rule,

$$p(s|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|s)p(s)}{p(\boldsymbol{x})}$$

  - And sum rule,

$$p(\boldsymbol{x}) = \sum_{s} p(\boldsymbol{x}|s)p(s)$$

Inferring from models

- Maximum a Posteriori (MAP) estimate
  - We can write,

$$p(s|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|s)p(s)}{\sum_c p(\boldsymbol{x}|s)p(s)}$$

  - The denominator is constant w.r.t. x, so we can ignore it
  - So we can finally write,

$$s^* = \underset{s}{\operatorname{argmax}}\, p(\boldsymbol{x}|s)p(s)$$

Inferring from models

- Illustration: the noisy binary transmission
  - We assume a noisy transmission of binary symbols,

$$X = S + \epsilon$$

  - Where *e* is additive Gaussian noise

$$\epsilon = \mathcal{N}(0, \sigma)$$

Inferring from models

- Maximum a Posteriori (MAP) estimate
  - Then we can simply write,

$$P_{X|S}(x|s=0) = \mathcal{N}(x|0,\sigma)$$

$$P_{X|S}(x|s=1) = \mathcal{N}(x|1,\sigma)$$

  - If we assume equal priors for the two binary symbols,

$$P_S(s=0) = P_S(s=1) = 1/2$$

Inferring from models

- The MAP estimate computed is the log domain is,

$$s^*(x) = \underset{s}{\operatorname{argmax}} \ \log P_{X|S}(x|s) + \log P_S(s)$$

- The prior probability is the same for all, then can further be ignored,

$$s^*(x) = \underset{s}{\operatorname{argmax}} \ \log P_{X|S}(x|s)$$

- After some maths developments,

$$s^*(x) = \underset{c}{\operatorname{argmin}} \ \frac{(x-\mu_c)^2}{\sigma^2}$$

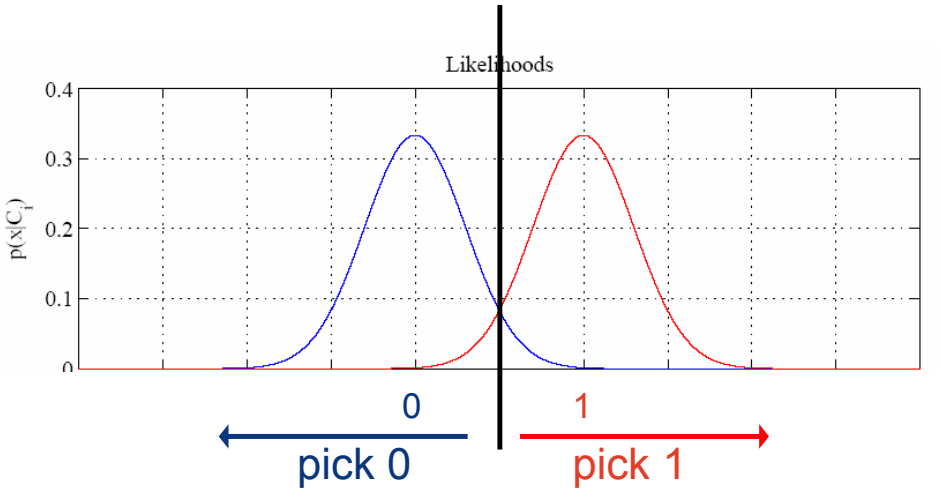
Inferring from models

MAP Gaussian 1-D

– Finally, the MAP decision is 0 if and only if,

$$(x-\mu_0)^2 < (x-\mu_1)^2$$

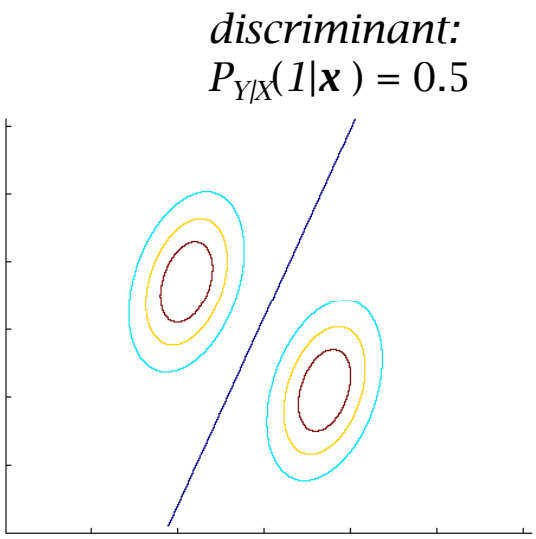
$$x < \frac{\mu_0+\mu_1}{2}$$

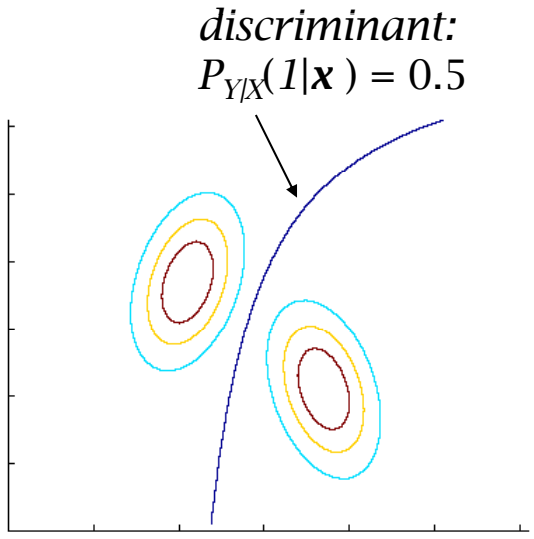


Inferring from models

MAP Gaussian 2-D

- On the left: 2-D Gaussian noise with same covariance matrices, **linear decision boundary**
- On the right: 2-D Gaussian noise with different covariance matrices, **non-linear decision boundary**





Learning the model

Now, how do we learn the model (from an incomplete set of data)?

Maximum Likelihood Estimate (ML or MLE)

- 1-Gaussian distribution (for illustration)
- The **likelihood** of the data given the model is,

$$p(x_1, \ldots, x_N | \theta = \{\mu, \sigma\})$$

Learning the model

Maximum Likelihood Estimate (ML or MLE)

- The objective of the learning problem is to solve,

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmax}}\, p(x_1, \ldots, x_N|\theta)$$

- This can be rewritten as an optimization problem (assuming a differentiable and convex function),

$$\widehat{\theta} = \underset{\theta}{\operatorname{argmax}}\, p(x_1, \ldots, x_N|\theta) \rightarrow \frac{\partial}{\partial\theta}\, p(x_1, \ldots, x_N|\theta) = 0$$

Learning the model

Maximum Likelihood Estimate (ML or MLE)

- Assuming the data are **i.i.d.** (independent and identically distributed),

$$p(x_1, \ldots, x_N|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

- The **log-likelihood** can be written as,

$$\log p(x_1, \ldots, x_N|\theta) = \sum_{n=1}^{N} \log p(x_n|\theta)$$

- And the learning problem is equivalent to,

$$\sum_{n=1}^{N} \frac{\partial}{\partial \theta} \log p(x_n|\theta) = 0$$

Learning the model

Maximum Likelihood Estimate (ML or MLE)

- Assuming a Gaussian distribution,

$$\sum_{n=1}^{N} \frac{\partial}{\partial \theta} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} = 0$$

- Solving the equation w.r.t the mean,

$$\sum_{n=1}^{N} \frac{\partial}{\partial \mu} \log p(x_n|\mu,\sigma) = \cdots = \frac{1}{\sigma^2} \sum_{n=1}^{N} (x_n - \mu)$$

- Then (and also for the variance),

$$\widehat{\mu}^{MLE} = \frac{1}{N} \sum_{n=1}^{N} x_n \qquad \widehat{\sigma}^{MLE} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)^2}$$

Learning the model

Maximum Likelihood Estimate (ML or MLE)

- Generalization to N-D Gaussian distribution: straightforward (with some more matrixial operations)

  - Exercise

Learning the model

Maximum Likelihood Estimate (ML or MLE)

- Gaussian Mixture Model (GMM): not as simple
- **Auxiliary function** to maximize,

$$Q(\theta^j, \theta^{j+1}) = \sum_{n=1}^{N} \sum_{y} p(y|x_n, \theta^j) \ln p(y, x_n|\theta^{j+1})$$

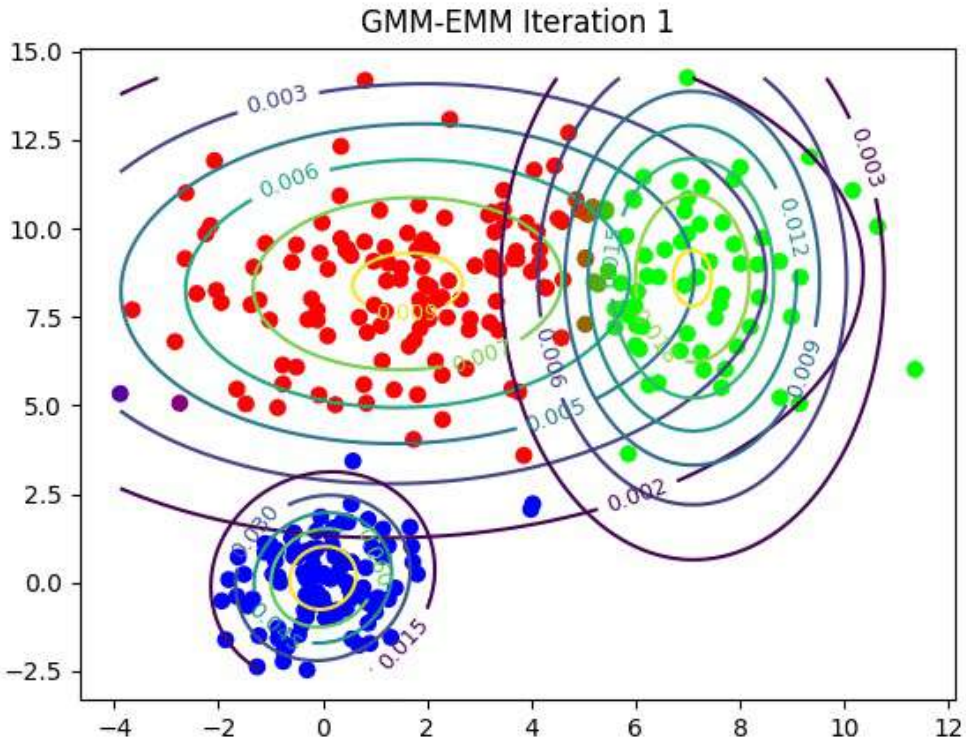
- **Expectation-Maximization** algorithm (EM)

$$\widehat{\alpha_k^{j+1}} = \frac{1}{\eta} \sum_{n=1}^{N} \gamma_{n,k}^j \qquad \mu_k^{j+1} = \frac{\sum_{n=1}^{N} \gamma_{n,k}^j x_n}{\sum_{n=1}^{N} \gamma_{n,k}^j} \qquad \Sigma_k^{j+1} = \frac{\sum_{n=1}^{N} \gamma_{n,k}^j (x_n - \mu_k)(x_n - \mu_k)^\top}{\sum_{n=1}^{N} \gamma_{n,k}^j}$$

Learning the model

Maximum Likelihood Estimate (ML or MLE)

- Until convergence !



SCIENCES
SORBONNE
UNIVERSITÉ

4.
Why is is so hard to learn?

Memorization vs. Generalization

The objective of machine learning is to learn to **generalize** to unseen data

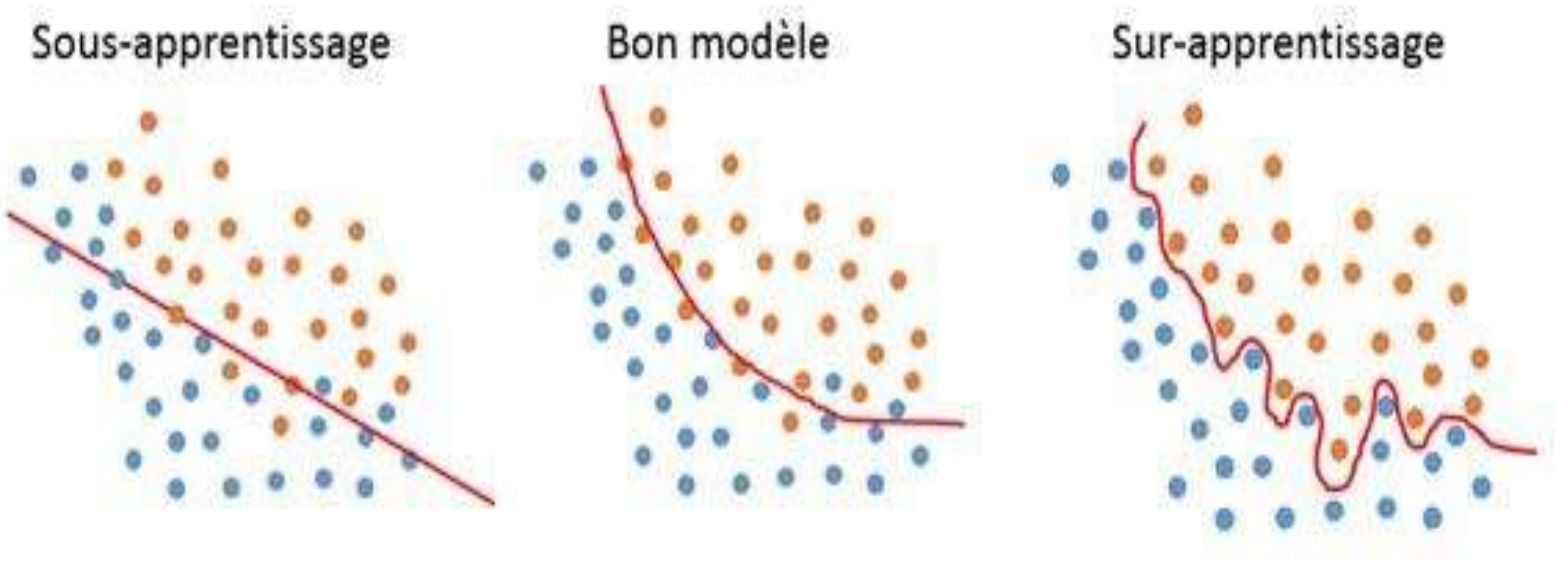
- «Learning by heart» is **memorizing** but **not learning**
- The loss on the training data set is not an indicator of the generalization ability of a model
- The problem of **over-fitting**
- «The **curse of dimensionality** »

Training set

New data

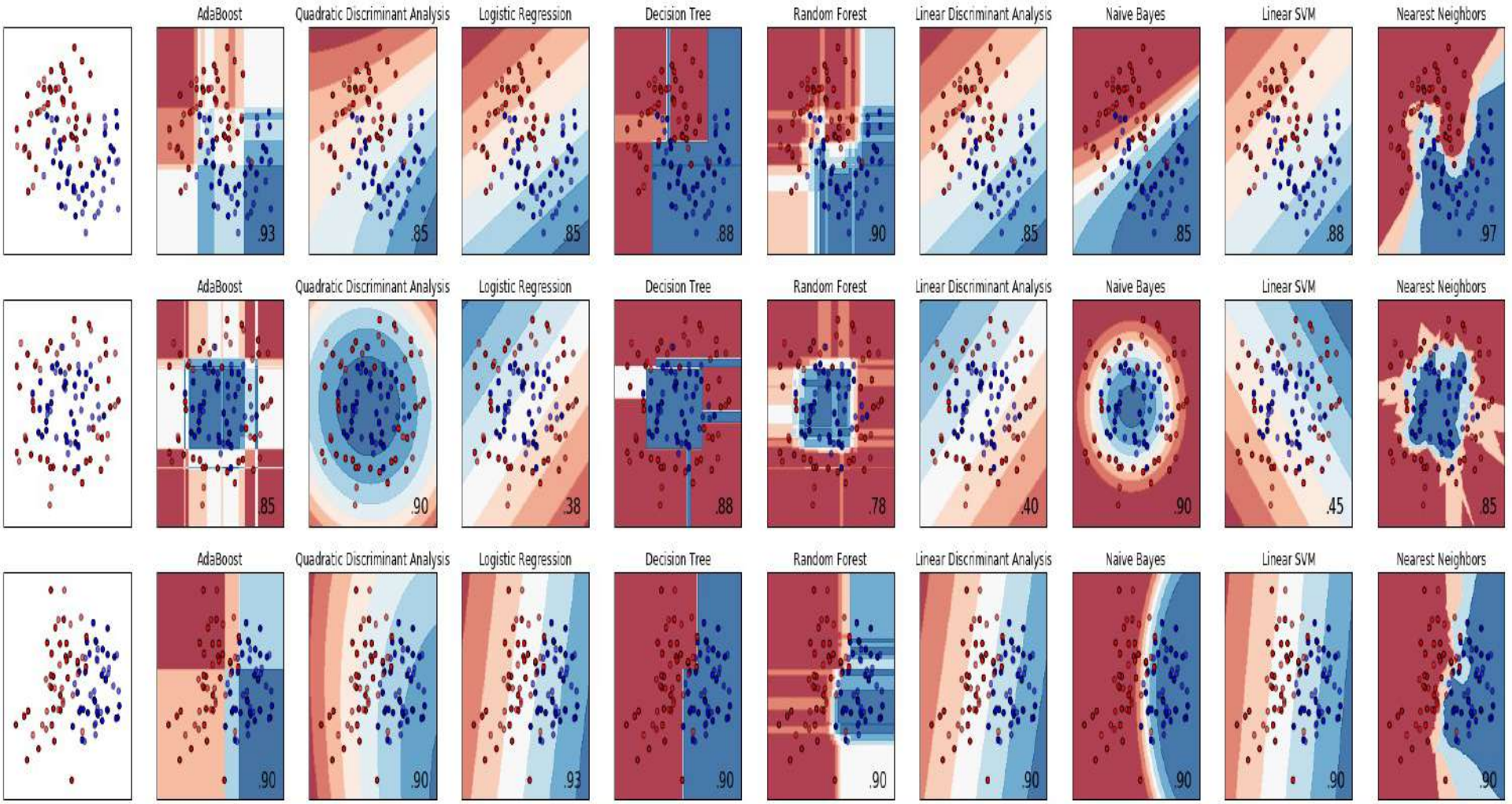
Decision boundary and incomplete data

The problem of overfitting



Decision boundary and incomplete data

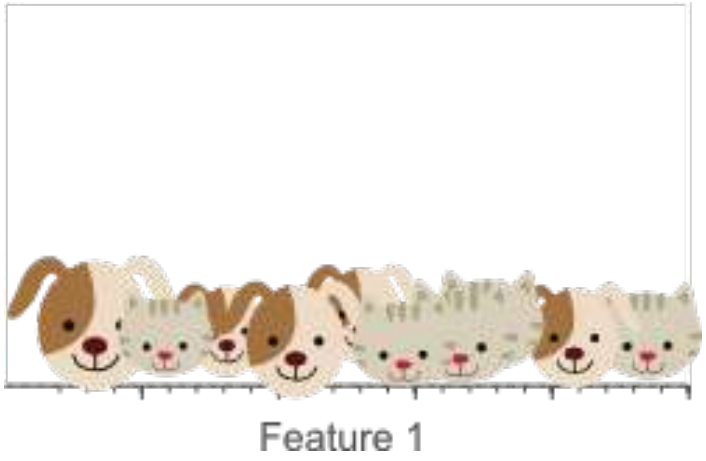
Visualization of decision boundaries



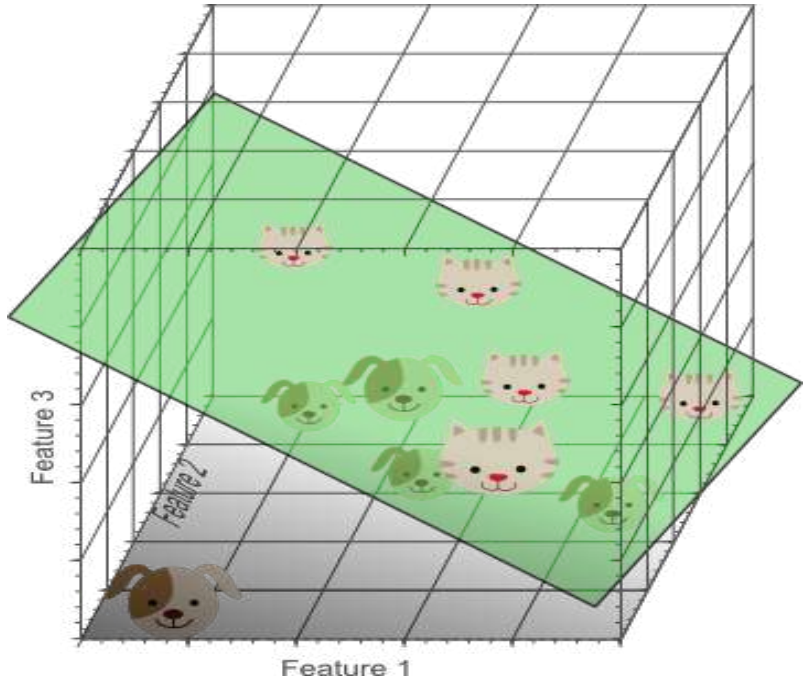
«The curse of dimensionality»

The dimensionality of the data space is a problem inherent to machine learning

- Augmenting the dimensionality of the data space can lead to a **better separation** of the data



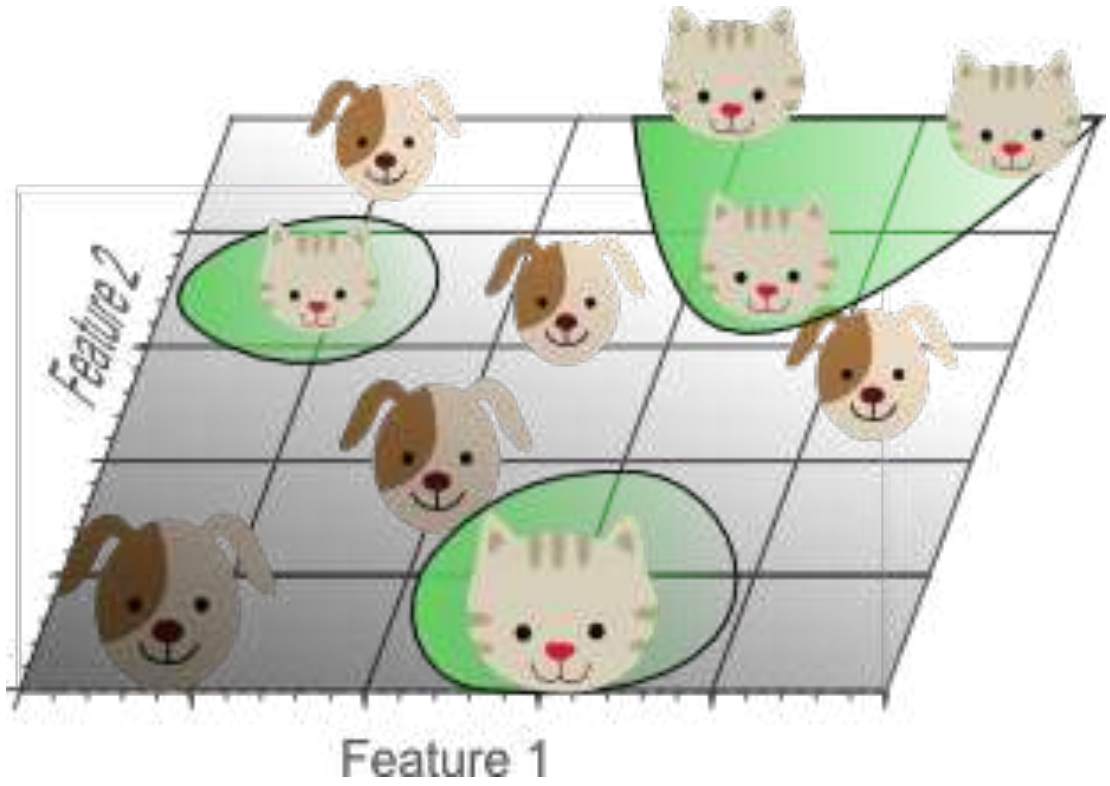
Bad separation in 1-D ☹



Good separation in 3-D ☺

«The curse of dimensionality»

- Which in turn increases the risk of isolation of the data samples, and reinforce the **risk of overfitting**

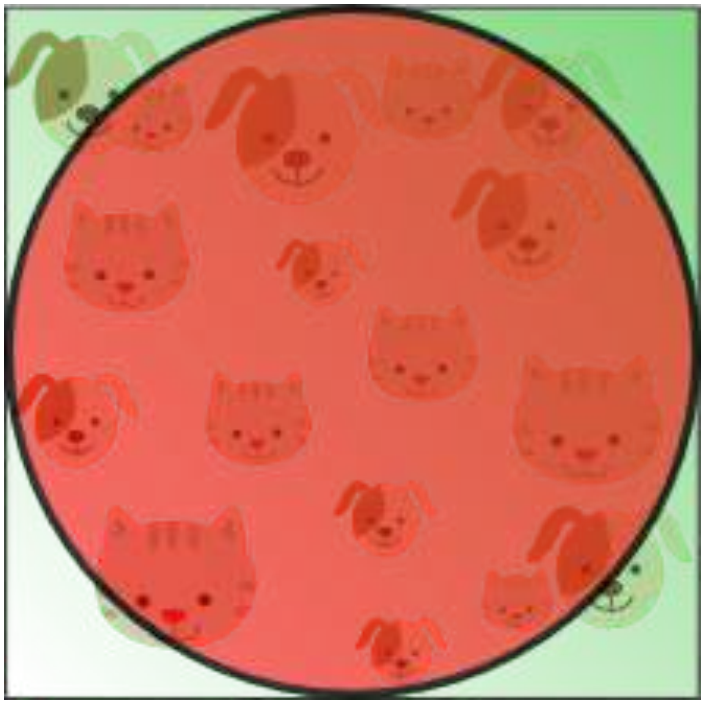


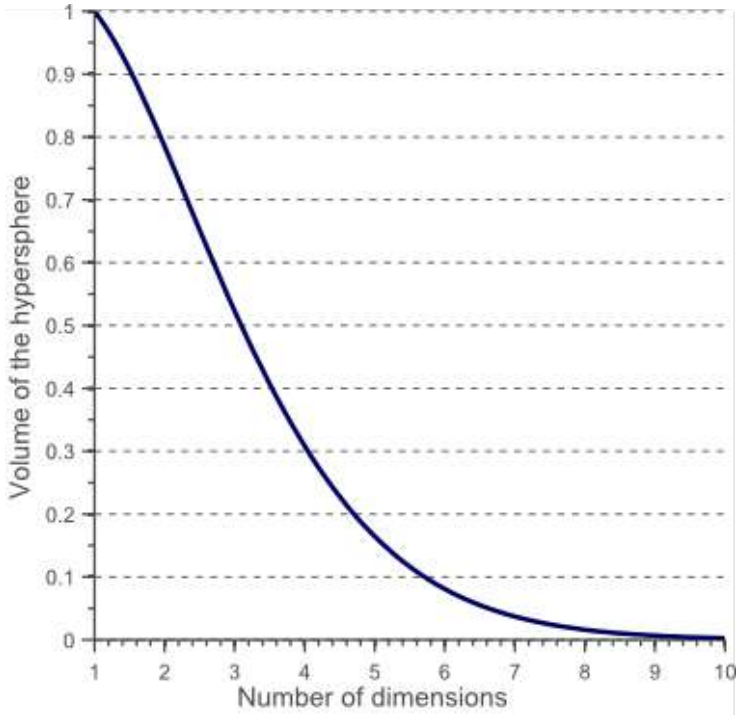
«The curse of dimensionality»

Wait, why???

Example of a hyper-sphere inscribed within a hyper-cube (e.g., in 2 dimensions)

The volume of the hyper-sphere tends toward 0 with the augmentation of the dimensionality!!!





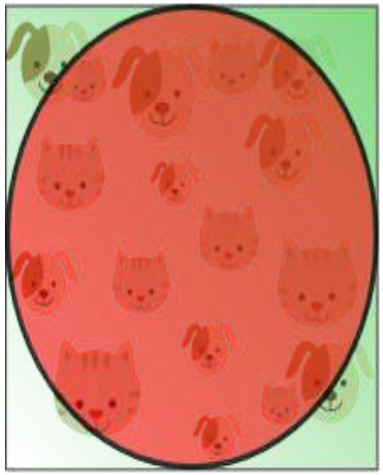
«The curse of dimensionality»

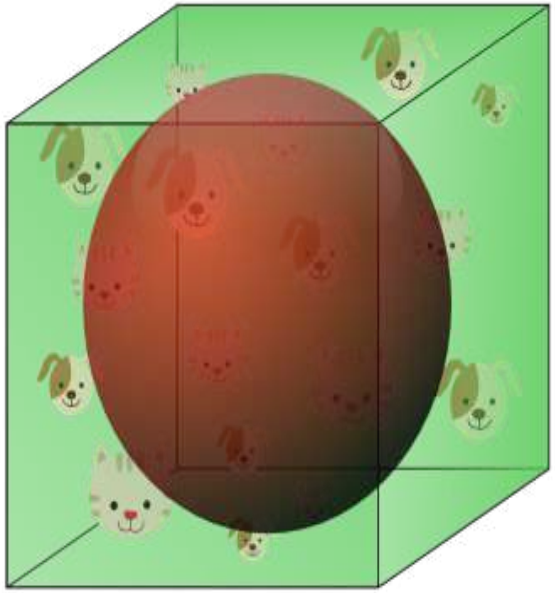
Wait, why???

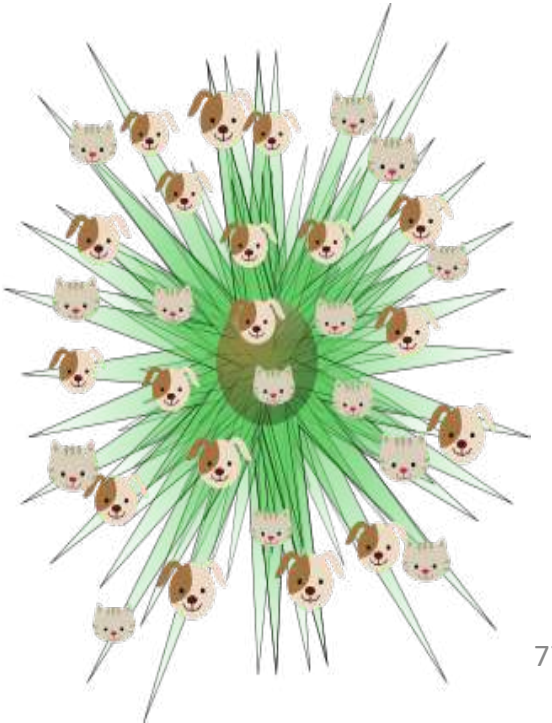
Samples go to the corners of the space

Then, they are all at equal distance from each others

Spherization of the space!!!







SCIENCES SORBONNE UNIVERSITÉ

5. Model validation and good practices

Model assessment

- Objectives
  - Asses whether a model will **generalize** to **independent** data (typically, new previously unseen data)
  - Can be proceed by means of **cross-validation** and **holdout**
  - Few and limited theoretical properties: asymptotical behaviour in some case of CV
  - …But experimentally essential !!!

Short reminder on random variables, expectation, estimator and bias

Definition 1 *The expectation of a random variable X with finite values in $\{x_1; \ldots; x_N\}$ is defined as,*

$$\mathbb{E}(X) = \sum_{i=1}^{N} p(X = x_i)\, x_i$$

where: $p(X = x_i)$ is the probability that X has the value x_i

Property 1 [Constant]: *The expectation of a constant K (w.r.t to X) is:*

$$\mathbb{E}_X(K) = K$$

Property 2 [Linearity]: *The expectation of λX is:*

$$\mathbb{E}_X(\lambda X) = \lambda\, \mathbb{E}_X(X)$$

Property 3 [Independent random variables]: *The expectations of N independent random variables is:*

$$\mathbb{E}(\prod_{n=1}^{N} X_n) = \prod_{n=1}^{N} \mathbb{E}(X_n)$$

In particular, if X and Y are two independent variables:

$$\mathbb{E}(XY) = \mathbb{E}_X(X)\ \mathbb{E}_Y(Y)$$

Short reminder on random variables, expectation, estimator and bias

Definition 2 *The moment of order N of a random variable X is defined as,*

$$\mathbb{E}(X^N)$$

Definition 3 *The bias of an estimator θ from a random variable X is defined as,*

$$\mathbb{E}_X(\theta) - \theta$$

The estimator is unbiased if its bias is null.

Property 4 *The estimator*

$$\frac{1}{N}\sum_{i=1}^{N} x_i$$

is an unbiased estimator of the mean of the i.i.d random variable X

Model assessment

- Problem positioning
  - Let's consider two random variables X and Y related by their joint distribution F such as:

$$Y = F(X)(+\epsilon)$$

  - X is the prediction variable, generally continuous (for instance, with values in $\mathbb{R}^p$
  - Y is the predicted variable, either with continuous values (regression, for instance in $\mathbb{R}$) or discrete values (classification, for instance in $\{0,1\}$)

Model assessment

- Problem positioning
  - Now, we sample from this distribution a finite set of data, called observations:

$$\mathcal{T} = \{(x_1, y_1), \ldots, (x_{N_\tau}, y_{N_\tau})\}$$

  - From this training set, we fit a model $\widehat{F}$ (minimizing a given loss), such that:

$$\widehat{Y} = \widehat{F}(X)$$

  - We want to assess the *true error of this model, i.e. its ability to generalize on previously unseen data*
  - *We assume that these data are statistically independent from those in* $\mathcal{T}$ *(the case if X is iid) but sampled from the same distribution* F *(aka same population)*

Generalization error

Definition 1 *The true error (aka test error or generalization error) of a model $\widehat{f}$ given a training set $\mathcal{T}$ is defined as:*

$$E_{\mathcal{T}} = \mathbb{E}_X \left[L(Y, \widehat{F}(X)) | \mathcal{T} \right]$$

where: L is the loss function used to measure the error between the actual value of Y and the predicted one $\widehat{Y} = \widehat{F}(X)$

Typical loss functions L are:

- Regression: quadratic loss (then MSE)

$$L(y, \widehat{y}) = (y - \widehat{y})^2$$

- Classification: misclassification loss (and misclassification rate)

$$L(y, \widehat{y}) = \mathbb{1}_{y \neq \widehat{y}}$$

In most cases, the true error cannot be measured since we don't have access to F (except in case of simulation) ☹

Apparent error

Estimator 1 : one could use the error measured on the training set as an estimation of the true error

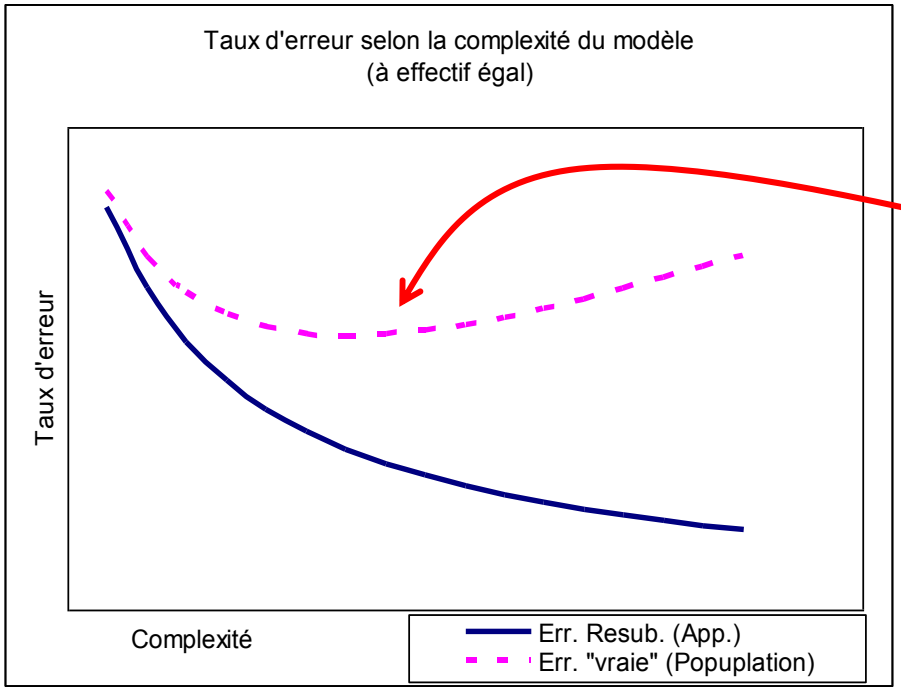
Definition 2 *The apparent error (aka train error) of a model $\hat{f}$ given a training set $\mathcal{T}$ is defined as:*

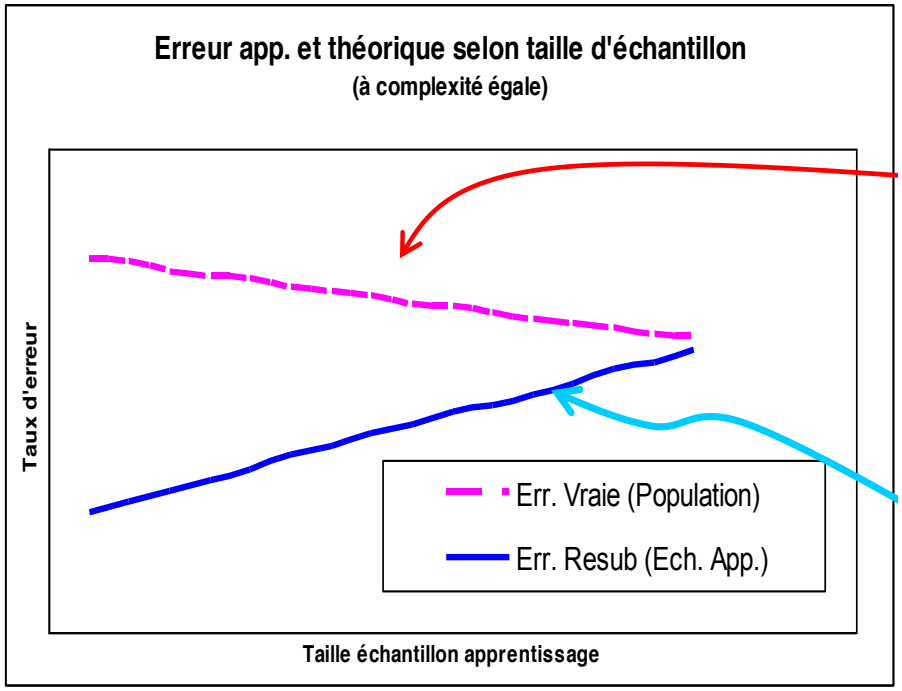
$$e_{\mathcal{T}} = \frac{1}{N_{\mathcal{T}}} \sum_{i=1}^{N_{\mathcal{T}}} L(y_i, \hat{F}(x_i))$$

- The apparent error is a **biased estimator of the true error**
  - ➢ the data used to estimate the error are not independent on those used to fit the model (see in-sample prediction error)

- This bias is generally optimistic, which means that the apparent error is lower (or much lower) than the true error.
- This is a typical case of over-fitting!

Apparent error





Hold-out

Estimator 2: one could split the set T into two subsets T' and t' so that,

$$\mathcal{T} = \mathcal{T}' \cup t' \text{ and } \mathcal{T}' \cap t' = \emptyset$$

- T' is used to fit the model
- t' is used to estimate the error on an independent set

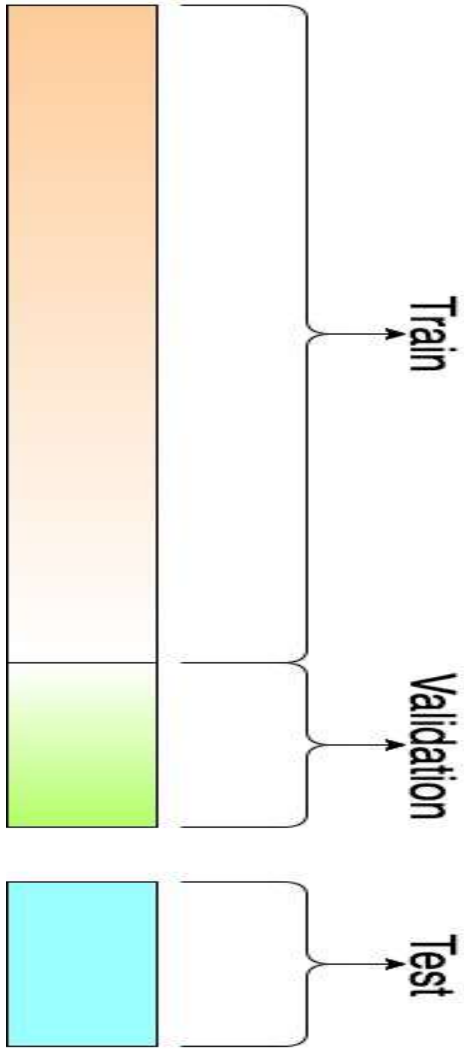
- t' is *hold-out* of the model fitting
- t' serves as *out-of-sample* data in order to estimate the true error

This estimator can be written as,

$$e'_t = \frac{1}{N_{t'}} \sum_{x \in t'} L(y, \widehat{F}_{\tau'}(x))$$

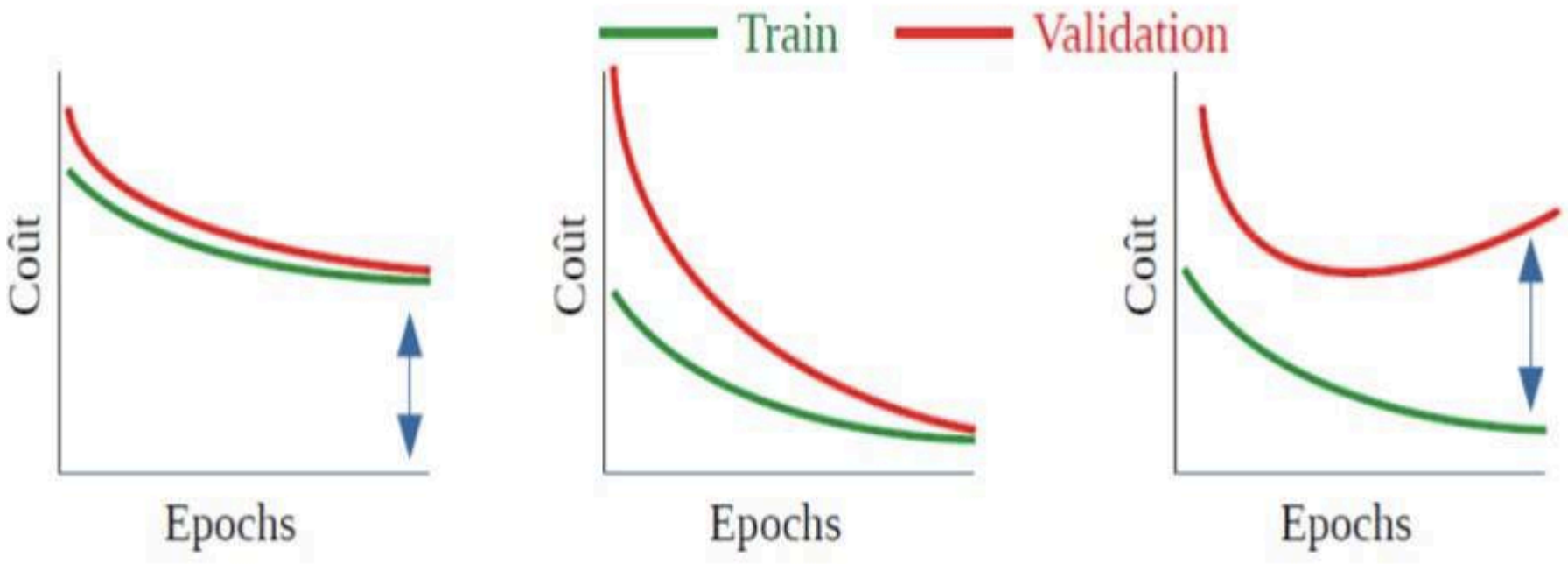
Hold-out

- Holdout
  - Use a single partition of the data into train / (validation) / and test set
  - Model's parameters are estimated on the train set
  - The validation set if used to monitor the model performance on unseen data, and for model selection
  - The performance of the selected model is measured on the test set



Hold out

- **Monitor** the performance on an independent validation set is essential to prevent **over-fitting!**
- Partition is generally about: 60%/20%/20%, 80%/10%/10%



Hold-out

Advantage

- an independent set is available to estimate the error

Drawback

- the train set T' is no longer the same as the original set T from which we want to estimate the true error (and is always smaller)

Properties (see: expectation and variance of an estimator)

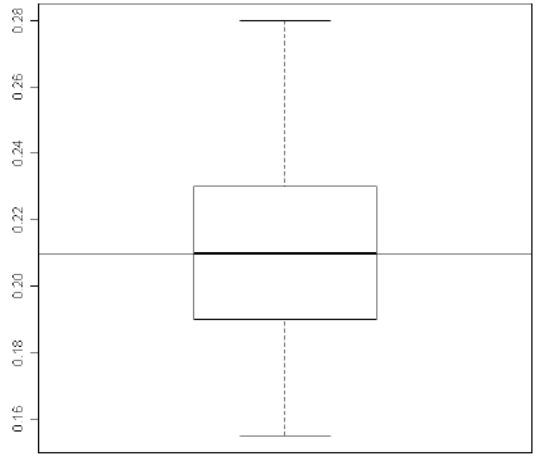
– $e_{t'}$ is an **unbiased estimator** of the true error of $E_{\mathcal{T}'}$ ☺

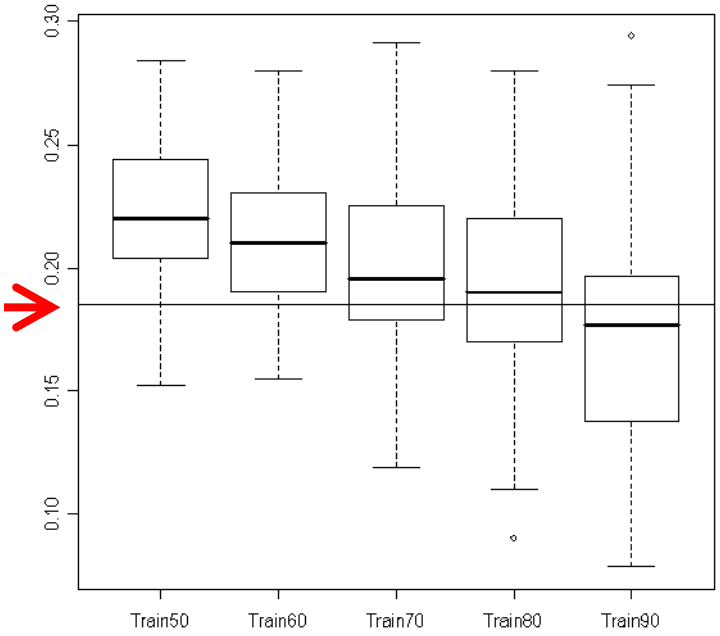
$$\mathbb{E}\left[e_{t'}\right] = E_{\mathcal{T}'}$$

– $var\left[e_{t'}\right]$ is proportional to $1/n'_t$

BUT $e_{t'}$ is a **biased estimator** of $E_{\mathcal{T}}$ ☹

Hold-out





Experimentation #1

- T = 500 samples
- T' = 300 samples, t' = 200 samples
- Repeating 100 times the fitting

➢ The error is unbiased (but has a certain variance from one repetition to the other)

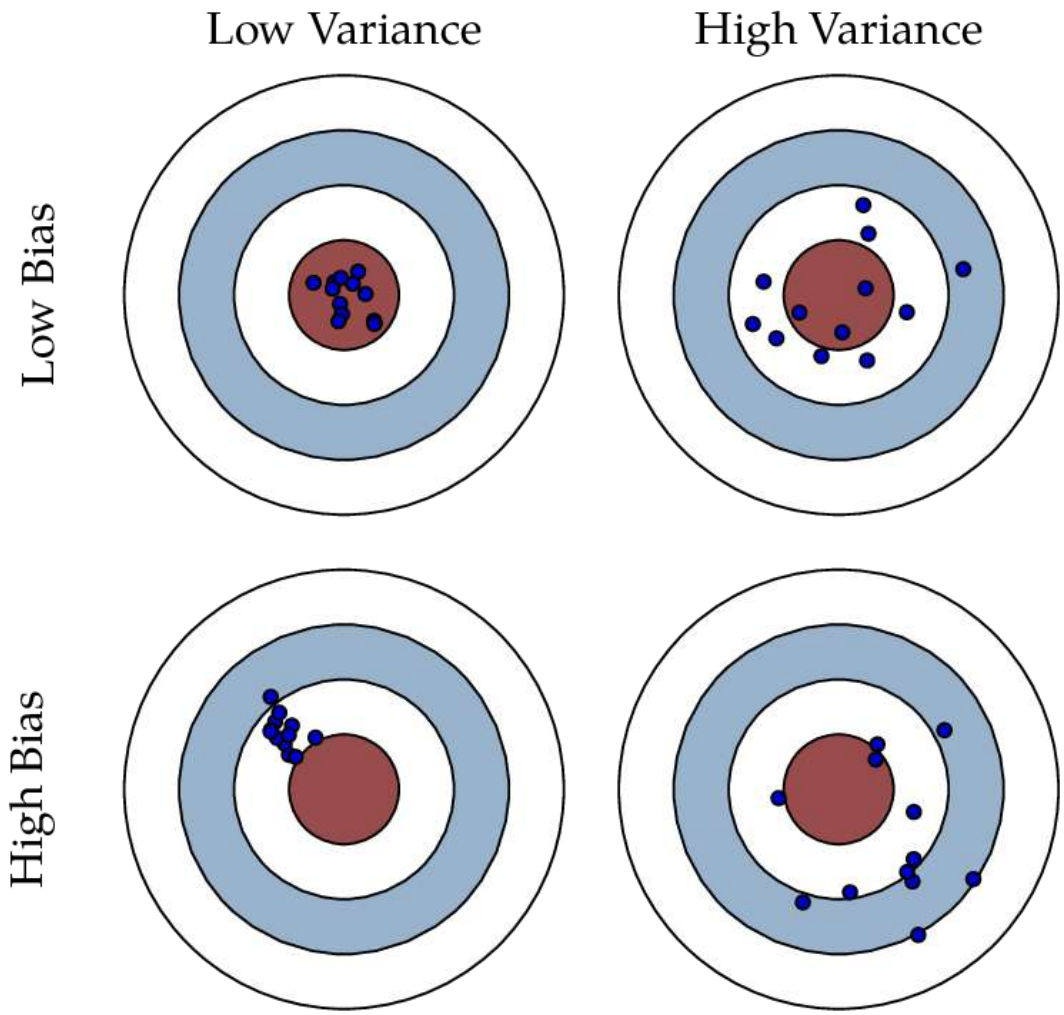
Experimentation #2

- Increasing the size of the train set

➢ Small: **high bias, low variance**
➢ Large: **low bias, high variance**

The bias-variance trade-off of the error!

Short reminder on bias and variance



Expected error

- Except with large datasets, the hold out is not a viable strategy to estimate the true error of a model
- This is generally the case for most real-world applications (though less and less with deep learning)

Instead, we define the **expected error** as.

Definition 3 *The expected error of a model $\widehat{F}$ at a point X' not included in τ is defined as:*

$$E(X') = \mathbb{E}_{\mathcal{T}}\left[L(Y, \widehat{F}_{\tau}(X))\right]$$

- This expression Is very similar to the definition of the true error, except that now the expectation is also computed across the possible T

Expected error

- In the following, we assume that the observed distribution is noisy

$$Y = F(X) + \epsilon$$

The noise variable ϵ is independent from X, with 0 mean and variance σ_ϵ

- Also, we assume a quadratic loss, so that the expected error can be written as,

$$\begin{aligned} E(X') &= \mathbb{E}_{\mathcal{T},\epsilon}\left[(Y - \widehat{F}_{\mathcal{T}}(X))^2\right] \\ &= \mathbb{E}_{\mathcal{T},\epsilon}\left[(F(X) + \epsilon - \widehat{F}_{\mathcal{T}}(X))^2\right] \end{aligned}$$

where the expectation is computed on $\mathcal{T}$ and ϵ

Bias/variance decomposition of the error

- By using a straightforward extension of the expression,

$$E(X') = \mathbb{E}_{\mathcal{T},\epsilon}\left[\epsilon^2\right] + \mathbb{E}_{\mathcal{T},\epsilon}\left[2\epsilon(Y - \widehat{F}_{\mathcal{T}}(X))\right] + \mathbb{E}_{\mathcal{T},\epsilon}\left[(Y - \widehat{F}_{\mathcal{T}}(X))^2\right]$$

- Considering the first term on the right side,

$$\mathbb{E}_{\mathcal{T},\epsilon}\left[\epsilon^2\right] = \sigma_\epsilon$$

- Considering the second term on the right side, and assuming statistical independence between X and ϵ ,

$$\begin{aligned}\mathbb{E}_{\mathcal{T},\epsilon}\left[2\epsilon(Y - \widehat{F}_{\mathcal{T}}(X))\right] &= 2\,\mathbb{E}_{\mathcal{T},\epsilon}\left[\epsilon\right]\; E_{\mathcal{T},\epsilon}\left[(Y - \widehat{F}_{\mathcal{T}}(X))\right] \\ &= 0\end{aligned}$$

Bias/variance decomposition of the error

- Now let's examine the last term,

$$\mathbb{E}_{\mathcal{T},\epsilon}\left[(Y-\widehat{F}_{\mathcal{T}}(X))^2\right] \;=\; \mathbb{E}_{\mathcal{T}}\left[(Y-\widehat{F}_{\mathcal{T}}(X))^2\right]$$

- Let's define,

$$\overline{F}(X)=\mathbb{E}_{\mathcal{T}}\left[\widehat{F}_{\mathcal{T}}(X)\right]$$

- Then,

$$\mathbb{E}_{\mathcal{T}}\left[(Y-\widehat{F}_{\mathcal{T}}(X))^2\right] \;=\; \mathbb{E}_{\mathcal{T}}\left[(Y-\overline{F}(X)+\overline{F}(X)-\widehat{F}_{\mathcal{T}}(X))^2\right]$$

- Again,

$$\begin{aligned}\mathbb{E}_{\mathcal{T}}\left[(Y-\widehat{F}_{\mathcal{T}}(X))^2\right] \;=\; & \mathbb{E}_{\mathcal{T}}\left[(Y-\overline{F}(X))^2\right]+\mathbb{E}_{\mathcal{T}}\left[(\overline{F}(X)-\widehat{F}_{\mathcal{T}}(X))^2\right] \\ & +2\;\mathbb{E}_{\mathcal{T}}\left[Y-\overline{F}(X)\right]\;\mathbb{E}_{\mathcal{T}}\left[\overline{F}(X)-\widehat{F}_{\mathcal{T}}(X)\right]\end{aligned}$$

Bias/variance decomposition of the error

- By definition,

$$\mathbb{E}_{\mathcal{T}}\left[\overline{F}(X) - \widehat{F}_{\mathcal{T}}(X)\right] = \overline{F}(X) - \mathbb{E}_{\mathcal{T}}\left[\widehat{F}_{\mathcal{T}}(X)\right] = 0$$

- Then,

$$\mathbb{E}_{\mathcal{T}}\left[(Y - \widehat{F}_{\mathcal{T}}(X))^2\right] = \mathbb{E}_{\mathcal{T}}\left[(Y - \overline{F}(X))^2\right] + \mathbb{E}_{\mathcal{T}}\left[(\overline{F}(X) - \widehat{F}_{\mathcal{T}}(X))^2\right]$$

- Finally,

$$E(X') = \sigma_\epsilon + \mathbb{E}_{\mathcal{T}}\left[(Y - \overline{F}(X))^2\right] + \mathbb{E}_{\mathcal{T}}\left[(\overline{F}(X) - \widehat{F}_{\mathcal{T}}(X))^2\right]$$

➢ This is the bias / variance decomposition of the error!

Bias/variance decomposition of the error

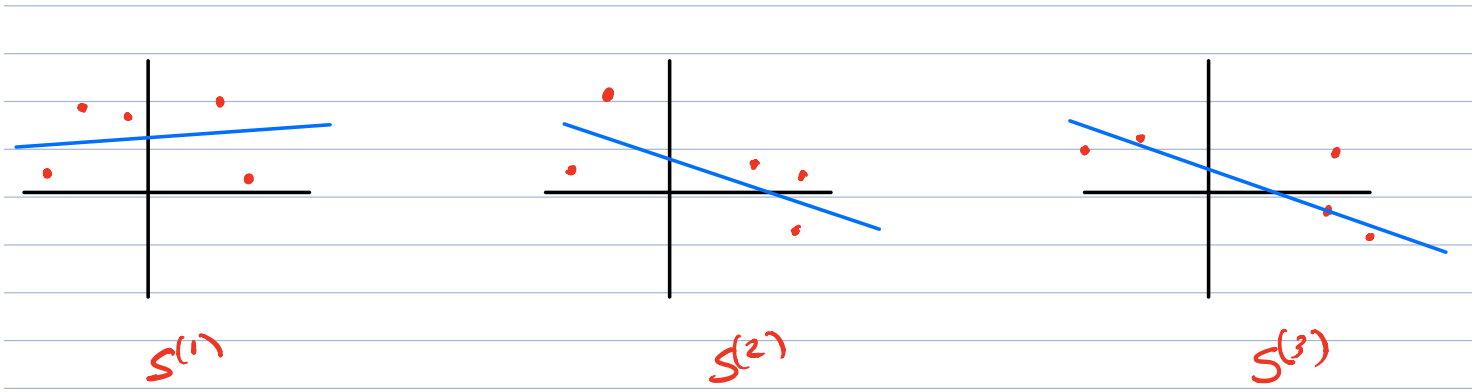
Understanding the three sources of error

- σ_ϵ is the **irreducible error**. This error represents the amount by which the observed data differ from the true ones. It does not depend on the training set or the model

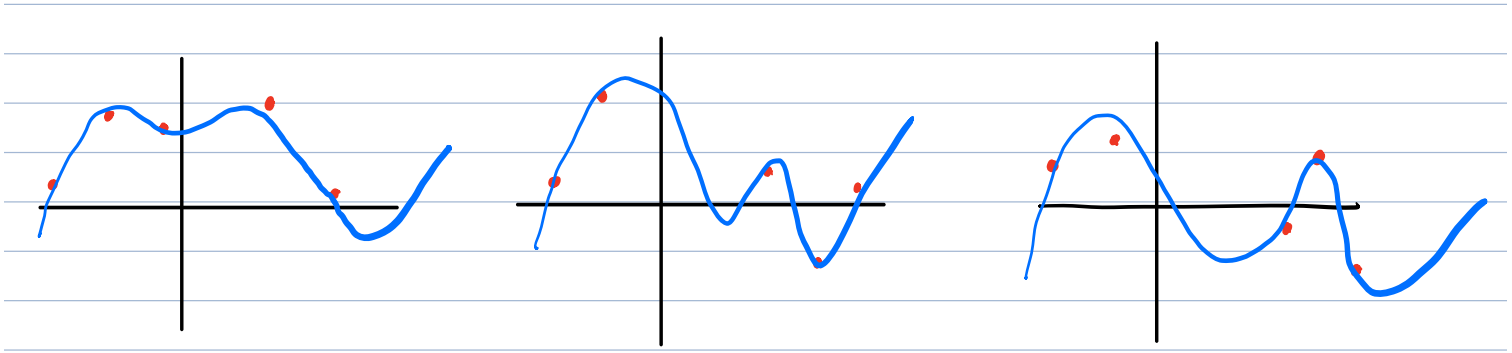
- $\mathbb{E}_{\mathcal{T}}\left[(Y - \overline{F}(X))^2\right]$ is the **squared bias**. This error represents the amount by which the mean of $\widehat{F}_{\mathcal{T}}(X)$ differs from the true $F(X)$

- $\mathbb{E}_{\mathcal{T}}\left[(\overline{F}(X) - \widehat{F}_{\mathcal{T}}(X))^2\right]$ is the **variance**. This error represents the amount by which $\widehat{F}_{\mathcal{T}}(X)$ differs from its own mean. This is caused by the variations of $\widehat{F}_{\mathcal{T}}(X)$ that are caused by small changes in the training set.

Bias/variance decomposition of the error



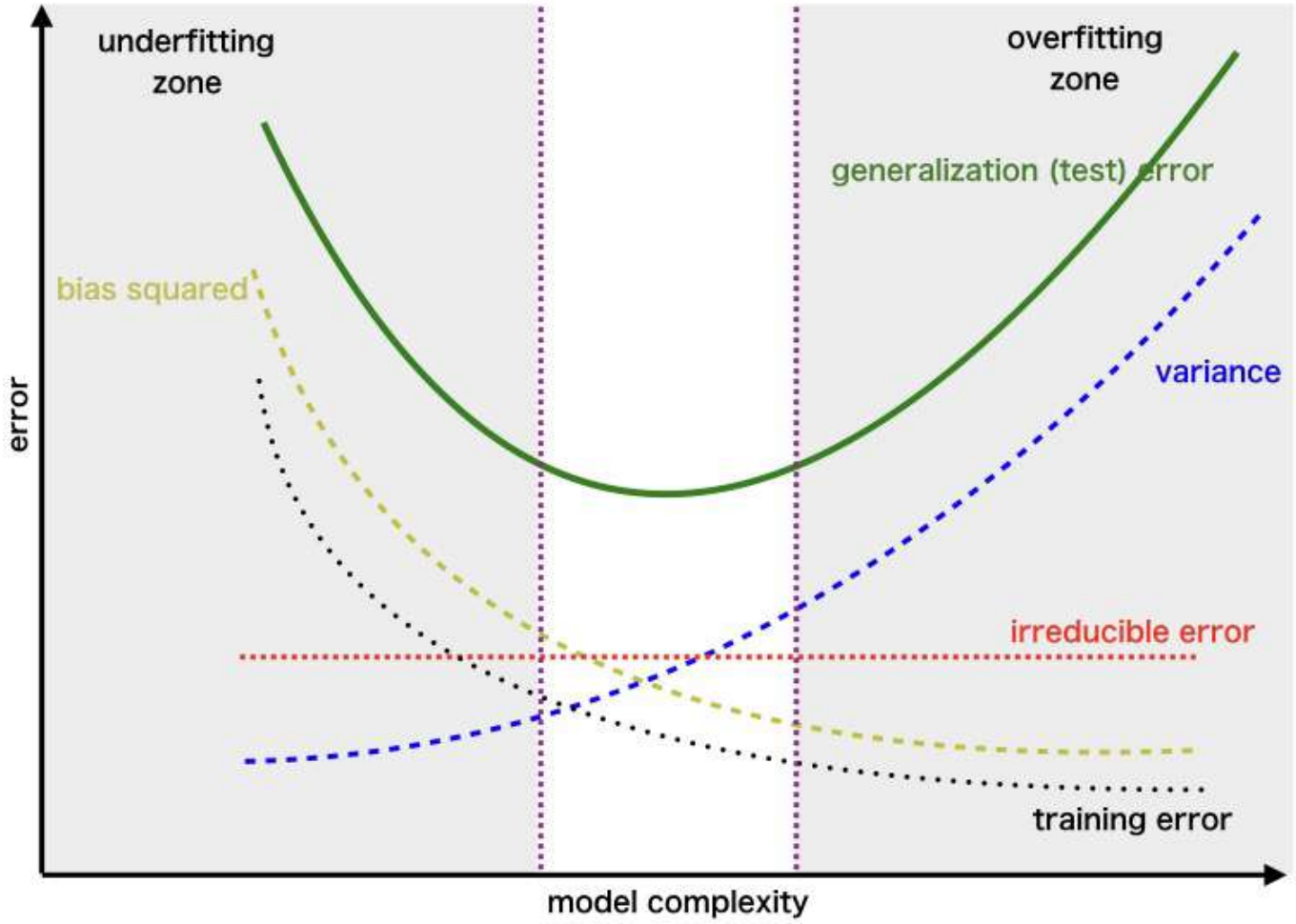
- Low complexity: high bias, low variance (under-fitting)



- High complexity: low bias, high variance (over-fitting)

Bias/variance decomposition of the error

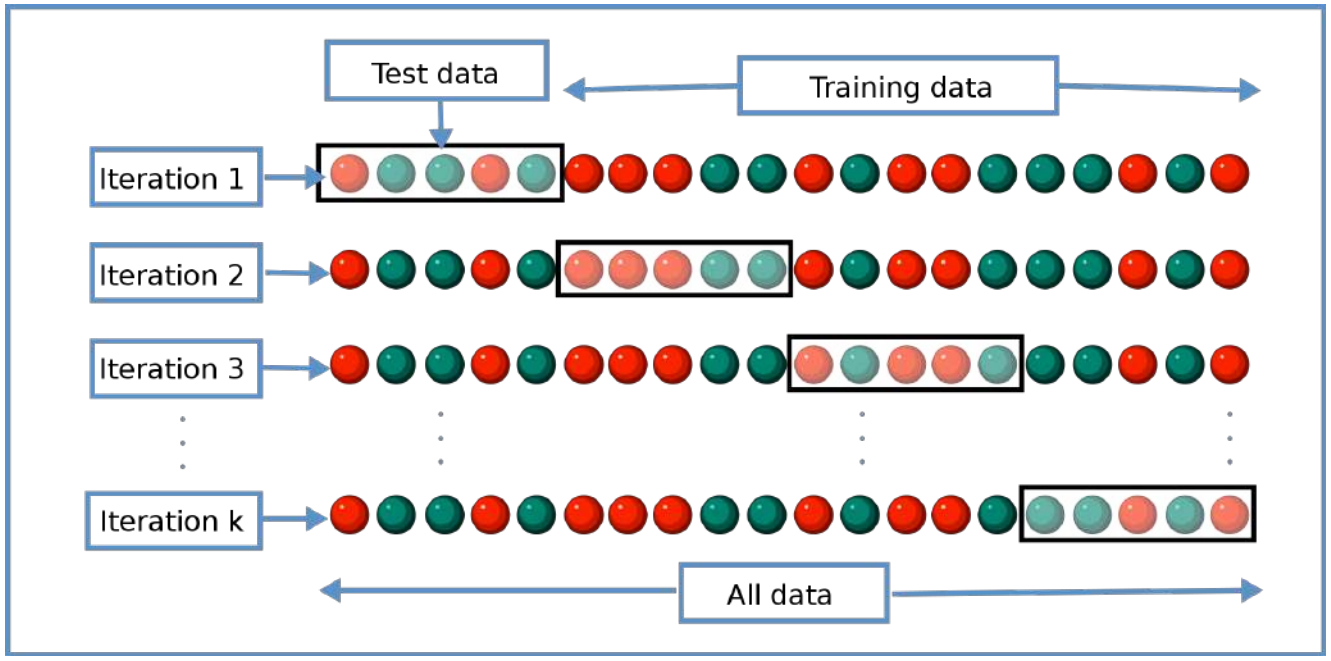
Summary and bias/variance trade-off



Cross-validation

Cross-validation (CV)

- Principle : repeated learning on various data sets
- Learn the model's parameters on the train set, and evaluate the performance on the test set
- Make permutations of train and test set, and repeat

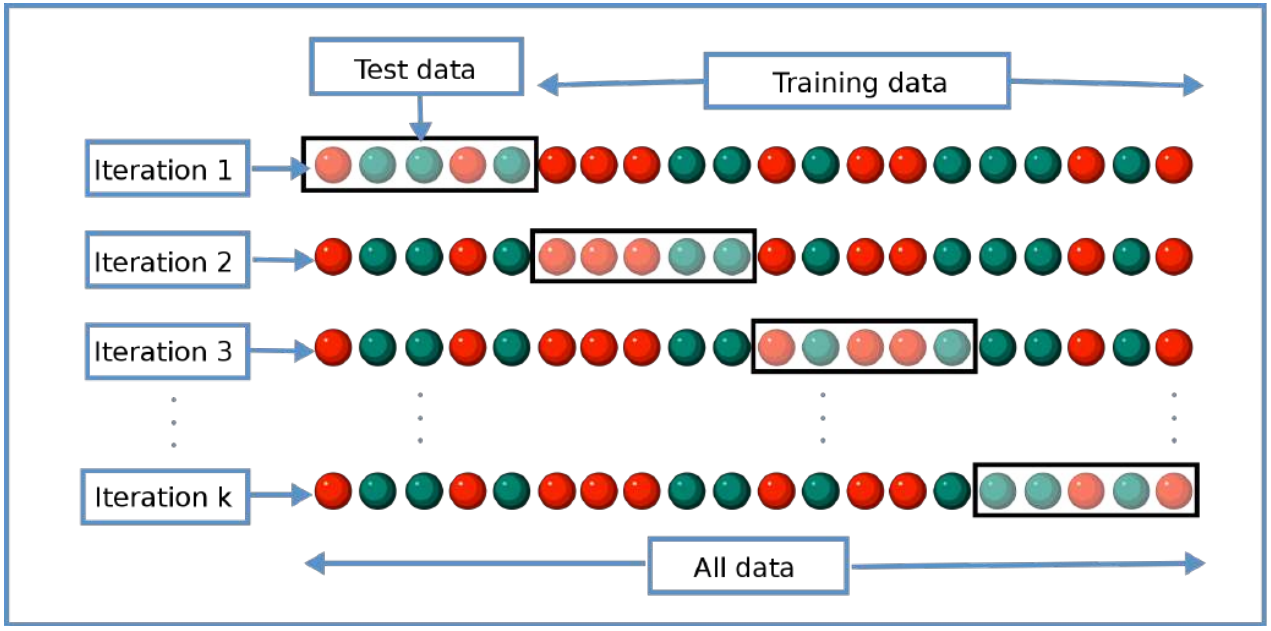


Cross-validation

Leave-p-Out CV

- Remove p samples (possibly p=1, aka the LOO) from the training set
- N – p samples in the train set
- p samples in the test set

➢ Repeat N / p times (can be extremely time-consuming!)

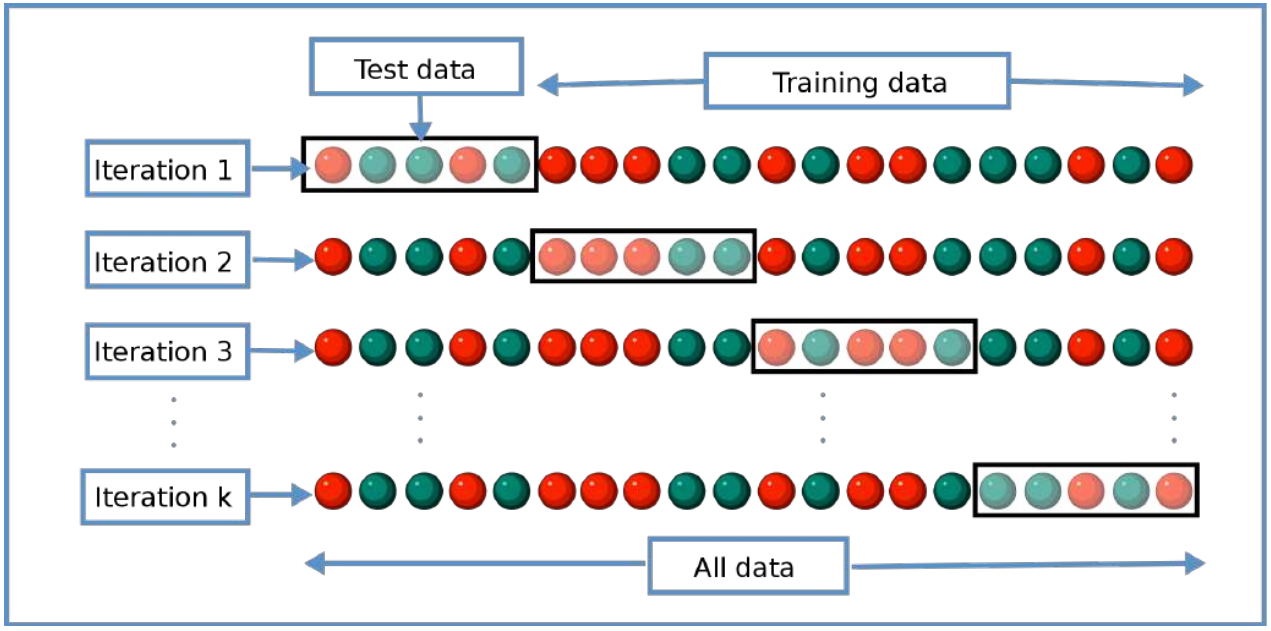


Cross-validation

K-fold CV

- Split data into K folds of equal sizes (K=2,5,10, 20, …)
- K-1 folds are used for training
- The remaining set is used for testing

➢ Repeat K times



Cross-validation

Properties

- The **empirical error** measured in CV is a good approximate of the expected error

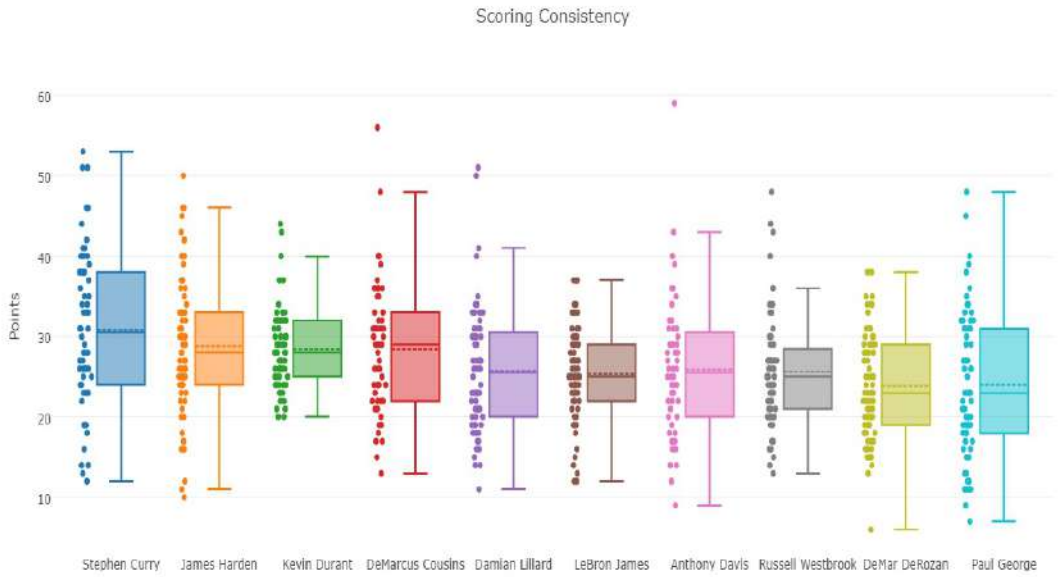
- The CV error is a **nearly unbiased estimator** of the expected error
  - ➢ The difference is due to the slight difference between the entire data set and the actual training sets

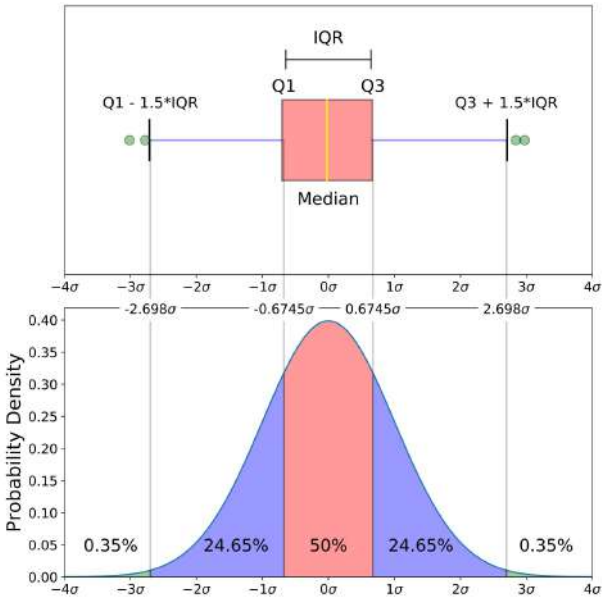
- Surprisingly, the LOO is also a nearly unbiased estimator of the expected error, **but not of the true error!** ☹
- Some other properties (especially wrt the variance) are more difficult to interpret, since the various training sets are clearly not independent from each others (high overlap between repetitions)

- Generally, K=10 offers a **good trade-off between bias and variance**

Cross-validation

- More about CV
  - CV allows to remove performance **bias** by estimating the **error distribution** of the model
  - Have an insight of the actual **expected error**
  - Post statistical comparison of models (e.g., ANOVA)





Cross-validation

- Stratified CV
  - Often, the class prior is unbalanced
  - To reduce the bias due to this cause, one can use stratified CV
  - Each fold has the same class distribution (reflecting the class priors)

Model assessment

CV

Pros

- Good theoretical properties (nearly unbiased estimation of the EE)
- Model comparison and selection

Cons

- Time-consuming
- Generally limited to small databases

Holdout

Pros

- Simple

Cons

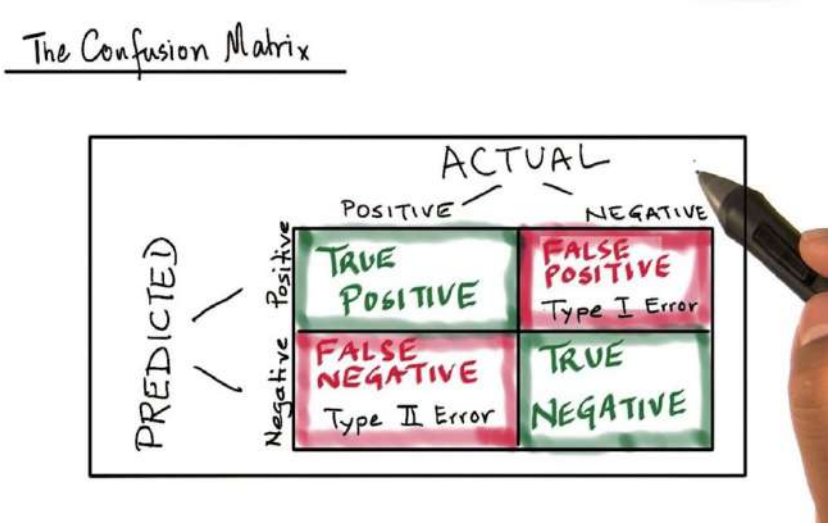
- Generally used for large data sets (e.g., in deep learning)
- Possibly subject to large variance or unknow behaviour!

Beware of hidden biases!!!

Measuring performance in classification tasks

- Binary classification

Precision: insertion rate,
recall: deletion rate,
and accuracy



$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

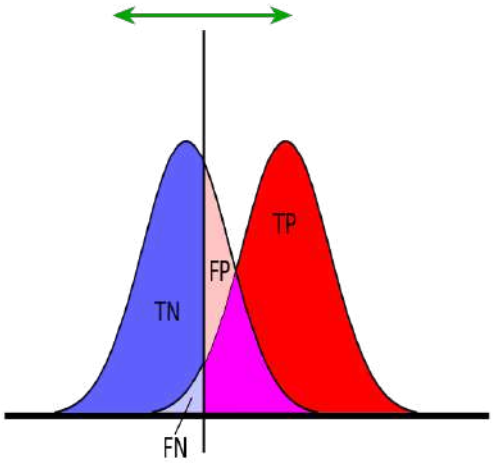
$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

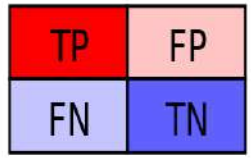
$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

Measuring performance in classification tasks

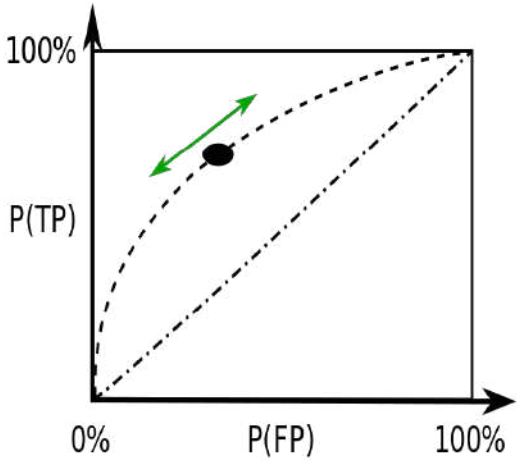
- ROC curve (*Receiver Operating Characteristics*)

- Illustrates the response of a binary classification/detection system as the **decision threshold** is varied



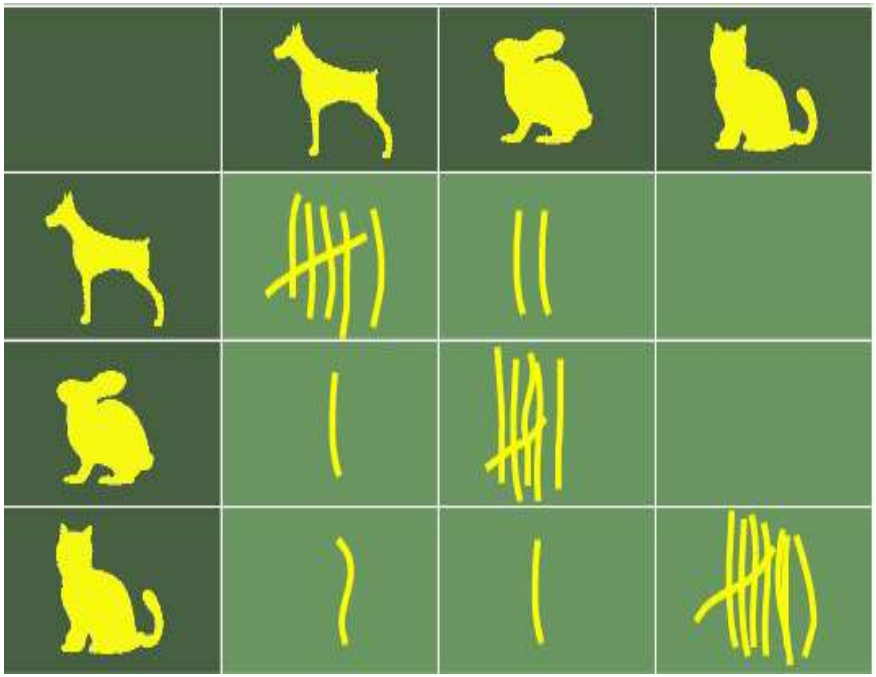


- AUC (Area under the curve) provides an overall measure of the performance
- The operating point can be selected as a **trade-off** between false positive and false negatives (depending on the task)
- One particular point: equal error rate **(EER)**



Measuring performance in classification tasks

- Multi-class



Confusion matrix for a 3-class problem

- Precision, recall, and accuracy per class (from the confusion matrix)
- Statistics: mean and variance (or confidence intervals)
- Global accuracy

- Unbalanced classes are subject to highly biased performance measures!!
- Balanced accuracy (binary classification) and its generalization to multi-class problems

$$BA = \frac{R_P + R_N}{2} = \frac{1}{2}\left(\frac{T_P}{T_P + F_N} + \frac{T_N}{F_P + T_N}\right) \quad (16)$$

where R_P and R_N are the positive and negative recalls, and T_P, T_N, F_P, and F_N are the true positive, true negative, false positive, and true negative counts.

Parameters and meta-parameters

– Parameters:

Free parameters that are estimated from data during learning

– Meta-parameters:

Parameters that are fixed by the user a priori

Example: Gaussian Mixture Models

- Parameters: priors, mean vectors, and covariance matrices
- Meta-parameters: number of component

Parameters and meta-parameters

Problem: how do we select the optimal number of components?

- The log-likelihood monotically increases with the number of components…
- So what???

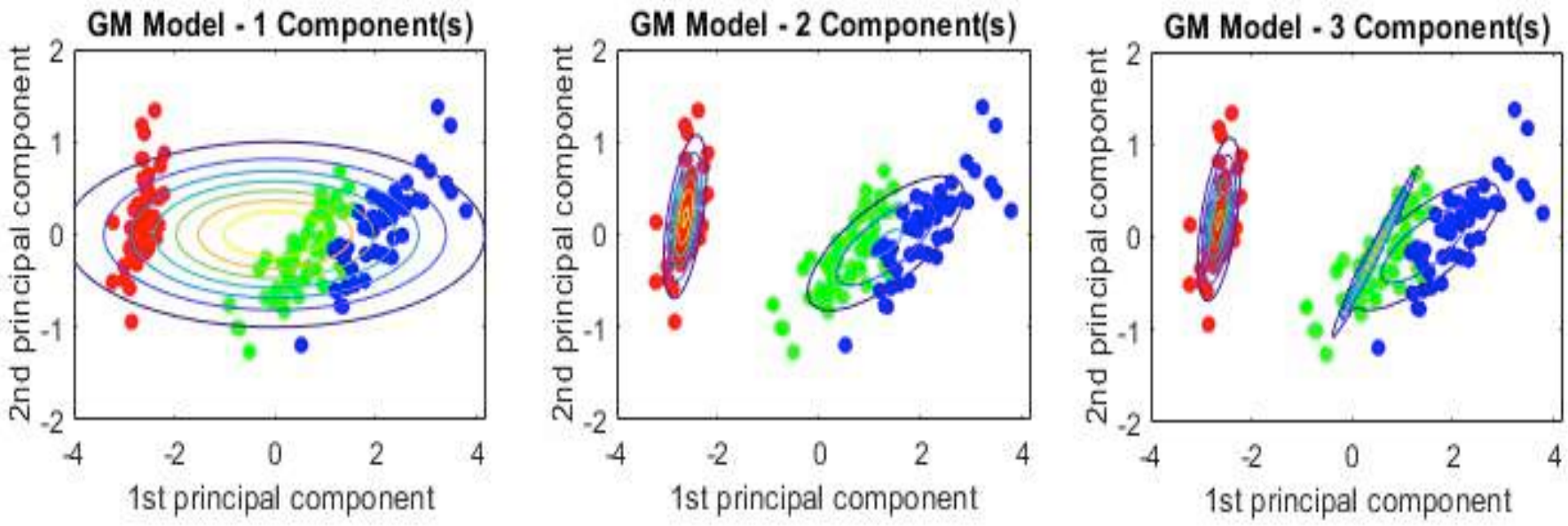
More generally, the more **complex** the model is, the better the model will fit the training data…

… until **over-fitting**

- The use of an independent **validation set** is a good practice in order to prevent this problem
- Other techniques exist for **model selection** (regularization techniques based on penalizing model complexity)

Model selection

- A trade-off between **model performance** and **model complexity**, in order to prevent from over-fitting



C1. Introduction to Machine Learning

Advanced Machine Learning (MLA)

M2 Engineering of Intelligent Systems

& Advanced Systems and Robotics

2020-2021